# MEGSV86w32.dll

# MEGSV86x64.dll

Reference Manual

ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

Last Modified: 27/01/2020
Version of DLL: 1.30

# Content





ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

## Introduction

This manual describes the Windows Dynamic Link Libraries MEGSV86w32.dll (32-Bit) and MEGSV86x64.dll (64-Bit). Both DLLs use the same functions and data types; they differ only in the function parameter calling convention: *__stdcall* for 32-Bit and *__fastcall* for 64-Bit.

They can be used for the **serial** RS-232/UART or USB-CDC (VCOM) interfaces for the device models **GSV-8** and **GSV-6**. Not all functions are available for GSV-6 and vice-versa, see function descriptions. Also, the device behavior may be different, especially in regard to device parametrization. While the GSV-8 puts changed settings into effect immediately (with exception of communication interface parameters) and stores everything automatically in non-volatile memory, the GSV-6 may do so or may not - please refer to device manual and protocol specification.

Almost all functions need the fundamental parameter *int ComNo*[1]. By this parameter, also the particular ressouces allocated for a device are distinguished, so it's present even for functions without hardware access. Up to 256 devices can be handled; i.e. the range of *ComNo* is from 1 to 256.

Functions return a meaningful read parameter or a return code. There are three different return codes defined:

GSV_OK = **0**: Function succeeded (non-signalling)

GSV_TRUE = **1**: Function succeeded (signalling, e.g. condition request is true)

GSV_ERROR = **-1**:  Function did not succeed. In that case, more information on error type and reason can be retrieved by calling GSV86getLastErrorText.

Functions may send and receive command frames to the device hardware or may not. This is indicated unter "Device Access". If the device is accessed, the device command number(s) are indicated. In that case, more information may be available in the device protocol documenation (ba-gsvcom_en.pdf ).

However, it's not recommended to call functions with device access redundantly too often, in order not to overburden the communication lines. Apart from that, especially with GSV-6, when parametrizing a device, it's a good practice to first read the related parameter, then compare with the desired one. Only if both differ, write the desired parameter (it's because GSV-6 writes parameters directly to EEPROM, which is subject to write wear).

1 The only exception is GSV86dllVersion

# Functions by categories

## Basics: Connect / Activate / Release

GSV86actExt
GSV86activateExtended
GSV86release
GSV86dllVersion

## Measuring values, buffers, data flow

GSV86readMultiple
GSV86read

GSV86clearDLLbuffer
GSV86clearDeviceBuf

GSV86received
GSV86startTX
GSV86stopTX
GSV86isValTXpermanent
GSV86triggerValue

GSV86setValDataType
GSV86getValObjectInfo

GSV86getFrequency
GSV86setFrequency
GSV86getDataRateRange
GSV86setZero

GSV86readRawValue
GSV86readValueString

## General Sensor parametrization

GSV86readUserScale
GSV86writeUserScale

GSV86getInTypeRange
GSV86setInType (GSV-8)
GSV86getAllInTypesRanges
GSV86readAnalogOutScale (GSV-8)
GSV86writeAnalogOutScale (GSV-8)

GSV86loadSettings
GSV86storeSettings

## Error management

GSV86getLastErrorText
GSV86getLastProtocollError


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

GSV86getLastDeviceError
GSV86getValueError
GSV86eraseValueErrMemory
GSV86resetErrorState

## Device information and administration

GSV86getSerialNo
GSV86getSoftwareConfiguration
GSV86readHWversion
GSV86firmwareVersion

GSV86readDeviceHours
GSV86writeDeviceHours

GSV86getWriteAccess
GSV86switchBlocking

GSV86getIDlevel
GSV86setPassword
GSV86changePassword

GSV86getIsCmdAvailable
GSV86getInterfaceIdentity
GSV86resetDevice (GSV-6)
GSV86readDevCalInfo
GSV86getModelInfo

## Measuring value related, unit

GSV86getUnitNo
GSV86setUnitNo
GSV86getUnitText
GSV86setUnitText

GSV86getModeMaxMin
GSV86setModeMaxMin
GSV86getMaxMinValue
GSV86clearMaxMinValue

GSV86readUserOffset
GSV86writeUserOffset

GSV86setMeasValProperty

GSV86readZeroValue
GSV86writeZeroValue

GSV86readAutoZeroSetting
GSV86writeAutoZeroSetting
GSV86getAutoZeroProperty

GSV86setAutoZeroProperty

## Device state and mode

GSV86getTXmode
GSV86setTXmode
GSV86getMode
GSV86setMode

## Multi-axis sensor

GSV86getFTsensorActive
GSV86setFTsensorActive
GSV86setFTarrayToRead
GSV86readFTsensorCalValue
GSV86readFTsensorCalArray
GSV86writeFTsensorCalArray
GSV86writeFTsensorGeoOffsets
GSV86readFTsensorSerNo
GSV86writeFTsensorFromFile
GSV86getFTsensorCalArrayInfo
GSV86setFTsensorActiveCalArray
GSV86eraseFTsensorCalArray
GSV86readFTsensorCalArrExt
GSV86writeFTsensorCalArrExt

## Filters

GSV86getModeAfilterAuto
GSV86setModeAfilterAuto
GSV86readAnalogFilterCutOff
GSV86writeAnalogFilterCutOff

GSV86getModeNoiseCut
GSV86setModeNoiseCut
GSV86getNoiseCutThreshold
GSV86setNoiseCutThreshold

GSV86getDfilterOnOff
GSV86setDfilterOnOff
GSV86getDfilterInfo
GSV86getDfilterType
GSV86getDfilterCoeff
GSV86setDfilterParams
GSV86calcSetDfilterParams
GSV86simulateDfilter

## Analogue output

GSV86getAnalogOutType
GSV86setAnalogOutType
GSV86readAnalogOutOffset


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

GSV86writeAnalogOutOffset
GSV86readAnalogOutScale
GSV86writeAnalogOutScale
GSV86writeAoutDirect

## Digital I/O

GSV86getDIOtype
GSV86setDIOtype
GSV86getDIOdirection
GSV86setDIOdirection
GSV86getDIOlevel
GSV86setDoutLevel
GSV86getDoutThreshold
GSV86setDoutThreshold
GSV86getDoutInitLevel
GSV86setDoutInitLevel

## Communication Interface

GSV86getInterfaceIdentity
GSV86readBasicInterfSettings
GSV86readInterfaceSetting
GSV86readAllInterfSettings
GSV86writeInterfaceSetting
GSV86setInterfaceOnOff
GSV86writeInterfaceBaud
GSV86readCANsettings
GSV86setCANsettings
GSV86getCANonOff
GSV86setCANonOff

## TEDS sensors / 1-wire IC

GSV86getTEDSactive
GSV86getSensorPlugged
GSV86readTEDSentry
GSV86readFormattedTEDSList
GSV86readTEDSrawData
GSV86writeTEDSrawData
GSV86read1wire
GSV86write1wire
GSV86read1wireTemperatur

## Counter / Frequency Input

GSV86getCounterFreqMode
GSV86setCounterFreqMode

## GSV-6 BT Bluetooth processor

The GSV-6 BT (ITA) device has an additional processor (referred to as "BGscript"). To access these functions, call GSV86BTinitConfig first. Finalize the GSV86BT... function

access by calling GSV86BTexitConfig. See also: ba-gsv6bt-commands.pdf

- GSV86BTinitConfig
- GSV86BTexitConfig
- GSV86BTgetMaxPower
- GSV86BTsetMaxPower
- GSV86BTgetBTmode
- GSV86BTsetBTmode
- GSV86BTreadName
- GSV86BTwriteName
- GSV86BTreset
- GSV86BTgetGSVonOff
- GSV86BTsetGSVonOff
- GSV86BTreadBatteryVoltage
- GSV86BTsetDefault
- GSV86BTgetInType
- GSV86BTsetInType
- GSV86BTsetDigitalOut
- GSV86BTgetLoggerInterval
- GSV86BTsetLoggerInterval

## Data Logger and Real-Time Clock

- GSV86readRTCtime
- GSV86writeRTCtime
- GSV86readLoggerSettings
- GSV86writeLoggerSettings
- GSV86controlFileLog
- GSV86querySDfileSys
- GSV86openSDfileDir
- GSV86getSDfileInfo
- GSV86readSDfile
- GSV86readSDfileExt
- GSV86copyFromSDfile
- GSV86getLoggerAlarmInterval
- GSV86setLoggerAlarmInterval


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

## All Functions

1 unsigned long CALLTYP GSV86dllVersion (void)
2 int CALLTYP GSV86actExt (int ComNo)
3 int CALLTYP GSV86activateExtended (int ComNo, unsigned long Bit-rate, unsigned long BufSize, unsigned long flags)
4 int CALLTYP GSV86getLastProtocollError (int ComNo)
5 int CALLTYP GSV86getLastDeviceError (int ComNo, int Async)
6 int CALLTYP GSV86getLastErrorText (int ComNo, char *ErrText)
7 int CALLTYP GSV86clearDLLbuffer (int ComNo)
8 int CALLTYP GSV86getInterfaceIdentity (int ComNo, int StopPermTX, int *NumOfIntfDescr, int *ThisInterfNo, int *WriteProtect, int *MValDataType, int *MValPermanenTX, int *NumObjInMVframe, int *ThisDeviceModel, int *ThisProtocol)
9 int CALLTYP GSV86clearDeviceBuf (int ComNo)
10 int CALLTYP GSV86received (int ComNo, int Chan)
11 int CALLTYP GSV86read (int ComNo, int Chan, double *out)
12 int CALLTYP GSV86readMultiple (int ComNo, int Chan, double *out, int count, int *valsread, int *ErrFlags)
13 int CALLTYP GSV86startTX (int ComNo)
14 int CALLTYP GSV86stopTX (int ComNo)
15 int CALLTYP GSV86release (int ComNo)
16 int CALLTYP GSV86getSerialNo (int ComNo)
17 int CALLTYP GSV86getTXmode (int ComNo, int Index)
18 int CALLTYP GSV86setTXmode (int ComNo, int Index, unsigned long TXmode)
19 int CALLTYP GSV86isValTXpermanent (int ComNo)
20 int CALLTYP GSV86getWriteAccess (int ComNo)
21 int CALLTYP GSV86setValDataType (int ComNo, int Datatype)
22 int CALLTYP GSV86firmwareVersion (int ComNo)
23 int CALLTYP GSV86readUserScale (int ComNo, int Chan, double *Norm)
24 int CALLTYP GSV86writeUserScale (int ComNo, int Chan, double Norm)
25 int CALLTYP GSV86getMode (int ComNo)
26 int CALLTYP GSV86setMode (int ComNo, unsigned long Mode)
27 int CALLTYP GSV86getFTsensorActive (int ComNo)
28 int CALLTYP GSV86setFTsensorActive (int ComNo, int OnOff)
29 int CALLTYP GSV86getModeAfilterAuto (int ComNo)
30 int CALLTYP GSV86setModeAfilterAuto (int ComNo, int OnOff)
31 int CALLTYP GSV86getModeNoiseCut (int ComNo)
32 int CALLTYP GSV86setModeNoiseCut (int ComNo, int OnOff)
33 int CALLTYP GSV86getModeMaxMin (int ComNo)
34 int CALLTYP GSV86setModeMaxMin (int ComNo, int OnOff)
35 int CALLTYP GSV86getValObjectInfo (int ComNo, double *ScaleFactors, unsigned long *ObjMapping, int *DataType)
36 int CALLTYP GSV86getValMapping (int ComNo, int Index)

37 int CALLTYP GSV86setFTarrayToRead (int ComNo, int ArrNo)
38 int CALLTYP GSV86readFTsensorCalValue (int ComNo, int typ, int ix, double *val)
39 int CALLTYP GSV86readFTsensorCalArray (int ComNo, int ArrNo, char *SensorSerNo, double *MatrixNorm, double *InSens, double *Matrix, double *Offsets, double *MaxVals, double *Zvals)
40 int CALLTYP GSV86writeFTsensorCalArray (int ComNo, int ArrNo, const char *SensorSerNo, double MatrixNorm, double InSens, double *Matrix, double *Offsets, double *MaxVals, double *Zvals)
41 int CALLTYP GSV86writeFTsensorGeoOffsets (int ComNo, int ArrNo, double *offsets)
42 int CALLTYP GSV86readFTsensorSerNo (int ComNo, int ArrNo, char *SensorSerNo)
43 int CALLTYP GSV86writeFTsensorFromFile (int ComNo, int ArrNo, const char *DatFilePath)
44 int CALLTYP GSV86getFTsensorCalArrayInfo (int ComNo, int *MaxNumSupp, int *ArrNumStored)
45 int CALLTYP GSV86setFTsensorActiveCalArray (int ComNo, int ArrNo)
46 int CALLTYP GSV86eraseFTsensorCalArray (int ComNo)
47 double CALLTYP GSV86getFrequency (int ComNo)
48 int CALLTYP GSV86setFrequency (int ComNo, double frequency)
49 int CALLTYP GSV86setZero (int ComNo, int Chan)
50 int CALLTYP GSV86getInTypeRange (int ComNo, int Chan, double *Range)
51 int CALLTYP GSV86getAllInTypesRanges (int ComNo, int Chan, int *InTypes, double *Ranges)
52 int CALLTYP GSV86setInType (int ComNo, int Chan, int InType)
53 int CALLTYP GSV86readUserOffset (int ComNo, int Chan, double *Offset)
54 int CALLTYP GSV86writeUserOffset (int ComNo, int Chan, double Offset)
55 int CALLTYP GSV86loadSettings (int ComNo, int DataSetNo)
56 int CALLTYP GSV86storeSettings (int ComNo, int DataSetNo)
57 int CALLTYP GSV86getAnalogOutType (int ComNo, int Chan, int *Type)
58 int CALLTYP GSV86setAnalogOutType (int ComNo, int Chan, int Type, int Mode)
59 int CALLTYP GSV86writeAnalogOutOffset (int ComNo, int Chan, double Offset)
60 int CALLTYP GSV86readAnalogOutOffset (int ComNo, int Chan, double *Offset)
61 int CALLTYP GSV86readAnalogOutScale (int ComNo, int Chan, double *Scale)
62 int CALLTYP GSV86writeAnalogOutScale (int ComNo, int Chan, double Scale)
63 int CALLTYP GSV86writeAoutDirect (int ComNo, int Chan, int Code)
64 int CALLTYP GSV86getUnitNo (int ComNo, int Chan)
65 int CALLTYP GSV86setUnitNo (int ComNo, int Chan, int UnitNo)
66 int CALLTYP GSV86getUnitText (int ComNo, int Chan, int Code, char *UnitText)
67 int CALLTYP GSV86setUnitText (int ComNo, int Chan, int Code, char *UnitText)
68 int CALLTYP GSV86getDfilterOnOff (int ComNo, int Chan, int Type)
69 int CALLTYP GSV86setDfilterOnOff (int ComNo, int Chan, int Type, int OnOff)
70 int CALLTYP GSV86getDfilterInfo (int ComNo, int Chan, int TypeIn, int *TypeOut, double *CutOff)
71 int CALLTYP GSV86getDfilterType (int ComNo, int Chan, int TypeIn)
72 int CALLTYP GSV86getDfilterCoeff (int ComNo, int Chan, int Type, double *Coeff, double *CoeffB)
73 int CALLTYP GSV86setDfilterParams (int ComNo, int Chan, int Type, double *CutRatio, double *Coeff, double *CoeffB)
74 int CALLTYP GSV86calcSetDfilterParams (int ComNo, int Chan, int Type, double CutOff, double CutOffHi)
75 int CALLTYP GSV86simulateDfilter (int ComNo, int Analyze_Ftyp, double StartVal, double EndVal,

double Fa, int Points, char *Filepath)

76 double CALLTYP GSV86readDeviceHours (int ComNo, int Index)
77 int CALLTYP GSV86writeDeviceHours (int ComNo, double Hours)
78 int CALLTYP GSV86getNoiseCutThreshold (int ComNo, int Chan, double *Thres)
79 int CALLTYP GSV86setNoiseCutThreshold (int ComNo, int Chan, double Thres)
80 int CALLTYP GSV86getSoftwareConfiguration (int ComNo)
81 int CALLTYP GSV86setMeasValProperty (int ComNo, int PropType)
82 int CALLTYP GSV86getValueError (int ComNo, int Ix, int *ErrInfo, double *ErrTime)
83 int CALLTYP GSV86eraseValueErrMemory (int ComNo)
84 int CALLTYP GSV86resetErrorState (int ComNo)
85 int CALLTYP GSV86readAnalogFilterCutOff (int ComNo, double *CutOffFreq)
86 int CALLTYP GSV86writeAnalogFilterCutOff (int ComNo, double CutOffFreq)
87 int CALLTYP GSV86readZeroValue (int ComNo, int Chan)
88 int CALLTYP GSV86writeZeroValue (int ComNo, int Chan, int zero)
89 int CALLTYP GSV86getIDlevel (int ComNo)
90 int CALLTYP GSV86triggerValue (int ComNo)
91 int CALLTYP GSV86getMaxMinValue (int ComNo, int Chan, double *MaxValue, double *MinValue)
92 int CALLTYP GSV86clearMaxMinValue (int ComNo, int Chan)
93 int CALLTYP GSV86getIsCmdAvailable (int ComNo, int CmdUp, int CmdLo)
94 int CALLTYP GSV86switchBlocking (int ComNo, int OnOff)
95 int CALLTYP GSV86changePassword (int ComNo, char *NewPW)
96 int CALLTYP GSV86setPassword (int ComNo, char *password)
97 int CALLTYP GSV86getDIOdirection (int ComNo, int DIOgroup)
98 int CALLTYP GSV86setDIOdirection (int ComNo, int DIOgroup, int Direction)
99 int CALLTYP GSV86getDIOtype (int ComNo, int DIOno, int *AssignedChan)
100 int CALLTYP GSV86setDIOtype (int ComNo, int DIOno, int DIOtype, int AssignedChan)
101 int CALLTYP GSV86getDIOlevel (int ComNo, int DIOno)
102 int CALLTYP GSV86setDoutLevel (int ComNo, int DIOno, int DIOlevel)
103 int CALLTYP GSV86getDoutThreshold (int ComNo, int DIOno, double *ThresUp, double *ThresDown)
104 int CALLTYP GSV86setDoutThreshold (int ComNo, int DIOno, double ThresUp, double ThresDown)
105 int CALLTYP GSV86getDoutInitLevel (int ComNo, int DIOno)
106 int CALLTYP GSV86setDoutInitLevel (int ComNo, int DIOno, int DOInitLevel)
107 int CALLTYP GSV86getDataRateRange (int ComNo, double *DrateMax, double *DrateMin)
108 int CALLTYP GSV86readCANsettings (int ComNo, int Index, int *setting)
109 int CALLTYP GSV86setCANsettings (int ComNo, int Index, int setting)
110 int CALLTYP GSV86getCANonOff (int ComNo, int *CANappProt)
111 int CALLTYP GSV86setCANonOff (int ComNo, int OnOff)
112 int CALLTYP GSV86readInterfaceSetting (int ComNo, int Ix, int *Next, unsigned long *Data, int *ApplEnum, int *Writable)
113 int CALLTYP GSV86readBasicInterfSettings (int ComNo, int ActIntf, int *PhysEnums, int *ApplEnums, int *Flags)
114 int CALLTYP GSV86readAllInterfSettings (int ComNo, int IntNo, int *IntfEnums, int *Dtypes, int *Data,

int *BdList, int *BdNum)

115 int CALLTYP GSV86writeInterfaceSetting (int ComNo, int Ix, unsigned long Data)

116 int CALLTYP GSV86setInterfaceOnOff (int ComNo, int IntNo, int OnOff)

117 int CALLTYP GSV86writeInterfaceBaud (int ComNo, int IntNo, int Baud)

118 int CALLTYP GSV86readHWversion (int ComNo, int *MainHW, int *ExtHW)

119 int CALLTYP GSV86readTEDSentry (int ComNo, int Chan, int TemplID, int PropID, int *Next, int No, unsigned long *Udata, double *DblData, int *Flags)

120 int CALLTYP GSV86readFormattedTEDSList (int ComNo, int Chan, const char *TEDSfilePath, char *ListOut, int ListSize, int Code, char *ExtListOut)

121 int CALLTYP GSV86readTEDSrawData (int ComNo, int Chan, unsigned char *DataOut, int NumBytes, int StartByteAdr)

122 int CALLTYP GSV86writeTEDSrawData (int ComNo, int Chan, const unsigned char *DataIn, int NumBits, int StartBitAdr)

123 int CALLTYP GSV86getSensorPlugged (int ComNo, int Chan, int *BridgeSensor, int *TEDScapable)

124 int CALLTYP GSV86getTEDSactive (int ComNo, int Chan)

125 int CALLTYP GSV86resetDevice (int ComNo)

126 int CALLTYP GSV86getCounterFreqMode (int ComNo, int CntNo, int index)

127 int CALLTYP GSV86setCounterFreqMode (int ComNo, int CntNo, int index, int mode)

128 int CALLTYP GSV86readRawValue (int ComNo, int Chan)

129 int CALLTYP GSV86BTinitConfig (int ComNo, int *BGversion)

130 int CALLTYP GSV86BTexitConfig (int ComNo)

131 int CALLTYP GSV86BTgetMaxPower (int ComNo, int BTmode, int *pwr)

132 int CALLTYP GSV86BTsetMaxPower (int ComNo, int BTmode, int pwr)

133 int CALLTYP GSV86BTgetBTmode (int ComNo, int *BTmode)

134 int CALLTYP GSV86BTsetBTmode (int ComNo, int BTmode)

135 int CALLTYP GSV86BTreadName (int ComNo, char *Name)

136 int CALLTYP GSV86BTwriteName (int ComNo, char *Name)

137 int CALLTYP GSV86BTreset (int ComNo, int ResetType)

138 int CALLTYP GSV86BTgetGSVonOff (int ComNo, int *OnOff)

139 int CALLTYP GSV86BTsetGSVonOff (int ComNo, int OnOff)

140 int CALLTYP GSV86BTreadBatteryVoltage (int ComNo, double *Voltage)

141 int CALLTYP GSV86BTsetDefault (int ComNo)

142 int CALLTYP GSV86BTgetInType (int ComNo, int chan, int *val)

143 int CALLTYP GSV86BTsetInType (int ComNo, int chan, int val)

144 int CALLTYP GSV86BTsetDigitalOut (int ComNo, int DIOno, int OnOff)

145 int CALLTYP GSV86BTgetLoggerInterval (int ComNo, int *interval)

146 int CALLTYP GSV86BTsetLoggerInterval (int ComNo, int interval)

147 int CALLTYP GSV86readRTCtime (int ComNo, int ix, unsigned char *time)

148 int CALLTYP GSV86writeRTCtime (int ComNo, int ix, unsigned char *time)

149 int CALLTYP GSV86readLoggerSettings (int ComNo, int index, unsigned long *value)

150 int CALLTYP GSV86writeLoggerSettings (int ComNo, int index, unsigned long value)

151 int CALLTYP GSV86controlFileLog (int ComNo, int control)

152 int CALLTYP GSV86querySDfileSys (int ComNo, int Ctrl, int DirLevel, char *Name, int *Flags)

153 int CALLTYP GSV86openSDfileDir (int ComNo, char *Name, int Flags)

154 int CALLTYP GSV86getSDfileInfo (int ComNo, int *DirLevel, int *Flags, unsigned long *Size, char *Changed)

155 int CALLTYP GSV86readSDfile (int ComNo, unsigned char *data)

156 int CALLTYP GSV86readSDfileExt (int ComNo, unsigned char *data, int bufSize)

157 int CALLTYP GSV86copyFromSDfile (int ComNo, char *DstPath, char *SrcPath)

158 int CALLTYP GSV86getLoggerAlarmInterval (int ComNo, int *IntervSec)

159 int CALLTYP GSV86setLoggerAlarmInterval (int ComNo, int IntervSec, int setMode)

160 int CALLTYP GSV86writeFTsensorCalArrExt (int ComNo, int Styp, double *MatrixB, int *Fact1ix, int *Fact2ix, char *ModelName)

161 int CALLTYP GSV86readFTsensorCalArrExt (int ComNo, int *Styp, double *MatrixB, int *Fact1ix, int *Fact2ix, char *ModelName)

162 int CALLTYP GSV86readAutoZeroSetting (int ComNo, int Type, int Chan, double *val)

163 int CALLTYP GSV86writeAutoZeroSetting (int ComNo, int Type, int Chan, double val)

164 int CALLTYP GSV86getAutoZeroProperty (int ComNo, int Type, int ix, int *prop)

165 int CALLTYP GSV86setAutoZeroProperty (int ComNo, int Type, int ix, int prop)

166 int CALLTYP GSV86readDevCalInfo (int ComNo, unsigned char *CalTime, char *OpName)

167 int CALLTYP GSV86readValueString (int ComNo, int ChanIx, int flags, char *Vstring)

168 int CALLTYP GSV86getModelInfo (int ComNo, int *ExtModel, int *PeriMCUver, int *reserved)

# Function Documentation

## int CALLTYP GSV86actExt (int *ComNo*)

Simplified Version of GSV86activeteExtended, with: Bitrate=CONST_BAUDRATE =115200 Bits/s, BufSize=CONST_BUFSIZE =48000, flags=0

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Comport to be opened |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if ComPort could not be opened or device was not found. If =GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 2

Device access: Yes, see GSV86activateExtended

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86activateExtended (int *ComNo*, unsigned long *Bitrate*, unsigned long *BufSize*, unsigned long *flags*)

Opens a (V)COM-port, allocates all necessary resources and checks, if device is present. Must be called first.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Comport to be opened |
| in | *Bitrate* | Bits per second. Allowed values: 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1250000, 2500000, 3500000, 1750000 |

| | | |
|---|---|---|
| | | Must match device setting with UART, RS232 and RS422; but can be any of above values with USB-CDC (value will be discarded) |
| in | *BufSize* | Size of the buffer, given in number of measuring values. In this buffer measuring values are stored by the reading thread of this DLL and from where values are read by GSV86read/GSV86readMultiple.<br>Size is in Measuring values per channel-object; e.g. with MappedObjectNum=8 and BufSize=30000, 8 buffers are allocated, and each has a capacity of 30000 measuring values. |
| in | *flags* | Opens with special settings, specified by this flags. Can be ORed together.<br>ACTEX_FLAG_HANDSHAKE: if set, opens the port with hardware-handshake (RTS-CTS), reserved<br>ACTEX_FLAG_WAIT_EXTENDED: waits longer for device-answer<br>ACTEX_FLAG_STOP_TX stops continuous data transmission |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if ComPort could not be opened or device was not found. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 4

Device access: Yes, CmdNo: 0x80, 0x01, 0x2B

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86calcSetDfilterParams (int *ComNo*, int *Chan*, int *Type*, double *CutOff*, double *CutOffHi*)

Calculates and writes parameters and coefficients for the additional digital FIR/IIR filter.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | -1,0..8: If =0: Set Filter On/Off for all channels. Else: get Filter Coefficients for specified channel<br>If =-1: Do not write filter parameters to device. DLL stores Coefficients internally, e.g. for analysis (see GSV86simulateDfilter) |
| in | *Type* | Type of filter. Bit7: =0: IIR, =1: FIR<br>For IIR only: Bits<6:4>: =0: Low pass, =1: high pass, =2: band pass, =3: band stop. FIR only: Bit 6=1: Comb, Bit6=0: Low pass<br>Bits<3:0>: Filter order (as by now, fixed=4 for IIR, 4..14 for FIR)<br>See constant macros below (FILT_TYPE_<I/F>IR_<LP/HP/BP/BS>) |
| in | *CutOff* | -3dB cutoff frequency. With types band pass and band stop, this is the lower cut off frequency. |
| in | *CutOffHi* | Higher -3dB cutoff frequency. Only used for types band pass and band stop. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Additional possible error codes:
DF_ERR_NO_CONVERGENCE 0x30000203 coefficient calculation failed to converge
DF_ERR_COEFF_SUM_TOOBIG 0x30000208 coefficients are too big

OrdinalNo: 98

Device access: Yes, if Chan > -1, CmdNo: 0x4E, 0x50, 0x7E


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86changePassword (int *ComNo*, char * *NewPW*)

Changes the device password (=User-ID).

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *NewPW* | Char-String to new user password. Must be 4 chars long. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

Precondition: Old password given with GSV86setPassword

OrdinalNo: 151

Device access: Yes, CmdNo: 0x58

date: 16.04.2015 12:12

Applicable devices: GSV-8

### int CALLTYP GSV86clearDeviceBuf (int *ComNo*)

Clears and resets the measuring value buffer of the device. Frames present in send queue are not destroyed.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 9

Device access: Yes, CmdNo: 0x25

date: 10-31-2014

Applicable devices: GSV-8

### int CALLTYP GSV86clearDLLbuffer (int *ComNo*)

Clears and resets the measuring value buffer of the dll reading thread.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if an error occurred. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 8

Device access: No

date: 10-31-2014

### int CALLTYP GSV86clearMaxMinValue (int *ComNo*, int *Chan*)

Resets the maximum and minimum values, that the device had stored, if the corresponding mode flag is set (see GSV86getMode)

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 0..8: Channel to clear maximum and minimum value. 0: clear for all channels |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 148

Device access: Yes (Cmd 0x3C)

date: 16.04.2015

Applicable devices: GSV-8, GSV-6

### unsigned long CALLTYP GSV86dllVersion (void)

Returns the DLLs version number.

**Returns:**

Version Number High in Bits<31:16>, Version number Low in Bits<15:0>
Example: Version 1.1.00.32: return value 0x00010001 = d65537

**Note:**

3rd and 4th number of version resource NOT part of return value

OrdinalNo: 1 Device access: No

date: 10-31-2014

### int CALLTYP GSV86eraseFTsensorCalArray (int *ComNo*)

Erases the last of the Force/Torque (Multi-axis) sensor parameter array, if several arrays are stored.
Precondition: Device password (=User-ID) given.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

The 1st array (ArrNo=0) can't be erased, so the minimum number of stored calibration arrays is 1

OrdinalNo: 166

Device access: Yes, CmdNo: 0x7F

date: 10-31-2014

Applicable devices: GSV-8


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

## int CALLTYP GSV86eraseValueErrMemory (int *ComNo*)

Erases the devices fault memory.
Precondition: Device password (=User-ID) given.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 128

Device access: Yes, CmdNo: 0x44

date: 09.04.2015 16:15

Applicable devices: GSV-8

## int CALLTYP GSV86firmwareVersion (int *ComNo*)

Returns the devices software version numbers.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Firmware version: Bits<31:16>: Higher number of firmware version
Bits<15:0>: Lower number of firmware version
or GSV_ERROR if function failed
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 26

Device access: Yes, CmdNo: 0x2B

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86getAllInTypesRanges (int *ComNo*, int *Chan*, int * *InTypes*, double * *Ranges*)

Reads all available analog input types and ranges.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Number of input channel, 1..10 |
| out | *Types* | If not NULL: pointer to array of int, where the existent input types are written to. Type definition: see GSV86setInType() (InType) |
| out | *Ranges* | If not NULL: pointer to array of double, where the existent input ranges (sensitivities) are written to. Meaning depends on corresponding InType, which has the same index in InTypes array. Array sizes should be MAX_INPUT_TYPES_NUM =10 |

**Returns:**

Number of existent input types or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 66

Device access: Yes, CmdNo: 0xA2

date: 11-18-2014

Applicable devices: GSV-8

### int CALLTYP GSV86getAnalogOutType (int *ComNo*, int *Chan*, int * *Type*)

Reads the active type setting for the analog output.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: Channel number of analog output to read type and mode. |
| out | *Type* | Pointer, where Analog output type enum (see GSV86setAnalogOutType) is written to. |

**Returns:**

Mode: Flag value:
Bit 0: AOUT_MODE_DIRECT =1 Output is on, but does NOT react on measuring values,
but is direcly settable with GSV86writeAoutDirect
AOUT_ACTIVE_M_VALUES =0 Output is on and follows the corresponding measuring value input.
Bit 1: =1: AOUT_MODE_OFF 2 Output is off and high-impedance.
Bit 2: =1: Alternate Input (Out-Channels 7 or 8 only: linked to Counter/Frequency input)
Or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 112

Device access: Yes, CmdNo: 0x0D

date: 10-31-2014

Applicable devices: GSV-8

### int CALLTYP GSV86getCANonOff (int *ComNo*, int * *CANappProt*)

Reads the on/off state of the CAN/CANopen field bus interface.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | **CANappProt.* | Pointer to value, where application protocol info is written to:<br>=0 for proprietary (ME-Systeme) CAN "serial" protocol (reserved)<br>=1 for standardized CANopen protocol |

**Returns:**

GSV_TRUE if CAN/CANopen is on (enabled), GSV_OK is it is off (disabled) or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 169

Device access: Yes, CmdNo: 0x8C

date: 02.12.2015 12:32

Applicable devices: GSV-8, GSV-6 (from Firmware Version 3.25)

### int CALLTYP GSV86getDataRateRange (int *ComNo*, double * *DrateMax*, double * *DrateMin*)

Reads the possible value range for the measuring data frequency. *DrateMax* value depends on many device parametrization settings.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | **DrateMax* | pointer to double value, where maximum data frequency is written to |
| out | **DrateMin* | pointer to double value, where minimum data frequency is written to |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 164

Device access: Yes, CmdNo: 0x63

date: 18.08.2015 14:10

Applicable devices: GSV-8

### int CALLTYP GSV86getDfilterCoeff (int *ComNo*, int *Chan*, int *Type*, double * *Coeff*, double * *CoeffB*)

Reads array(s) of coefficients of the digital FIR/IIR filter.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: Get Filter Coefficients for specified channel |
| in | *Type* | =0=FILT_TYPE_IIR: Get Filter Coefficients for the IIR filter.<br>=0x80=FILT_TYPE_FIR: Get Filter Coefficients for the FIR filter. |
| out | *Coeff** | Pointer to double array of min. size =8. With IIR filter, at the first 5 places the a coefficients a0..a4 will be stored, with a0 at index 0.<br>With FIR filter, the 8 FIR coefficients will be stored. |
| out | *CoeffB** | Pointer to double array of min. size =4. Only with IIR filter, at the first 4 places the b coefficients b0..b3 will be stored, with b0 at index 0. Can be NULL with FIR filter. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 94

Device access: Yes, CmdNo: 0x4F

date: 10-31-2014

Applicable devices: GSV-8, GSV-6: IIR only

### int CALLTYP GSV86getDfilterInfo (int *ComNo*, int *Chan*, int *TypeIn*, int * *TypeOut*, double * *CutOff*)

Reads basic settings of the digital FIR/IIR filter.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: Get Filter Info for specified channel |
| in | *TypeIn* | =0=FILT_TYPE_IIR: Get Filter Info for the IIR filter.<br>=0x80=FILT_TYPE_FIR: Get Filter Info for the FIR filter. |
| out | *TypeOut* | Type of filter. Bit7: =0: IIR, =1: FIR<br>Bits<6:4>, IIR: =0: Low pass, =1: high pass, =2: band pass, =3: band stop. FIR: Bit 6=1: Comb, =0: Low pass<br>Bits<3:0>: Filter order (as by now, 4 for IIR, 4..14 for FIR)<br>See constant macros in MEGSV86...h (FILT_TYPE_<I/F>IR_<LP/HP/BP/BS>) |

| | | |
|---|---|---|
| out | *CutOff** | Pointer to double array of min. size=2. At array index 0, the lower -3dB Cutoff<br>frequency will be written. With filter types Band pass and band stop,<br>at array index 1, the higher -3dB Cutoff frequency will be written. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

CutOff frequencies depend on data frequency. They are internally stored as ratio FcutOff/Fdata

OrdinalNo: 92

Device access: Yes, CmdNo: 0x8A, 0x4B, 0x4D

date: 10-31-2014

Applicable devices: GSV-8, GSV-6: IIR only

## int CALLTYP GSV86getDfilterOnOff (int *ComNo*, int *Chan*, int *Type*)

Reads the enabled/disabled state of the digital FIR/IIR filter.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | If =0: Get Filter On/Off of all channels. 1..8: Get Filter On/Off of specified channel |
| in | *Type* | =0=FILT_TYPE_IIR: Get on/off state for the IIR filter.<br>=0x80=FILT_TYPE_FIR: Get on/off state for the FIR filter. |

**Returns:**

On/Off state(s): If Chan=0: On/Off state in Bits<7:0>, Bit=1: Filter enabled, =0: disabled;
while bit 0 corresponds to channel 1, ..., bit 7 to channel 8 If Chan = 1..8: Get Filter on/off state of specified channel:
=1=GSV_TRUE: Filter enabled, =0=GSV_OK: Filter disabled
or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 88

Device access: Yes, CmdNo: 0x51

date: 10-31-2014

Applicable devices: GSV-8, GSV-6: IIR only

## int CALLTYP GSV86getDfilterType (int *ComNo*, int *Chan*, int *TypeIn*)

Reads type/class/length information of the digital FIR/IIR filter.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: Get Filter Type for specified channel |
| in | *TypeIn* | =0=FILT_TYPE_IIR: Get Filter Type for the IIR filter.<br>=0x80=FILT_TYPE_FIR: Get Filter Type for the FIR filter. |

**Returns:**

Type of filter. Bit7: =0: IIR, =1: FIR
Bits<6:4> IIR: =0: Low pass, =1: high pass, =2: band pass, =3: band stop
FIR: Bit 6: =1: Comb, =0: Low pass
Bits<3:0>: Filter order (as by now, 4 for IIR, 4..14 for FIR)

See constant macros above (FILT_TYPE_<I/F>IR_<LP/HP/BP/BS>)
or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 93

Device access: Yes, CmdNo: 0x4B

date: 10-31-2014

Applicable devices: GSV-8, GSV-6: IIR only

## int CALLTYP GSV86getDIOdirection (int *ComNo*, int *DIOgroup*)

Reads the direction (input/output) of the digital I/O line's group.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DIOgroup* | The 16 digital IO lines are assigned to 4 groups, whose data direction can only be changed for that group of 4 IO lines.<br>Assignment is as follows:<br>GroupNo DIOno BitNo in return-val, if DIOgroup=0<br>1 1..4 0<br>2 5..8 1<br>3 9..12 2<br>4 13..16 3<br>If DIOgroup is =0, all 4 directions are returned in Bits<3:0>, whereby 1=input and 0=output. |

**Returns:**

Data direction: GSV_TRUE (=1) for Input and GSV_OK (=0) for output
or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 152

Device access: Yes, CmdNo: 0x59

date: 18.06.2015

Applicable devices: GSV-8

## int CALLTYP GSV86getDIOlevel (int *ComNo*, int *DIOno*)

Reads the level (high/low) of the digital I/O line(s), independently of type and other settings.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DIOno* | Number of digital IO line 1..16. Or 0=: Get all 16 DIO levels in Bits<15:0> of return value, whereby bit0 = DIOno 1.. Bit15= DIOno 16 |

**Returns:**

DIO level: GSV_TRUE, if DIO level = high. GSV_OK, if level = low,
or Bits<15:0> = all levels, if DIOno=0. Or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). /note: DIO level is read independently of DIOtype or DIOdirection.

OrdinalNo: 155

Device access: Yes, CmdNo: 0x5D

date: 21.06.2015

Applicable devices: GSV-8, eventually GSV-6

### int CALLTYP GSV86getDIOtype (int *ComNo*, int *DIOno*, int * *AssignedChan*)

Reads the type of the digital I/O line.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DIOno* | Number of digital I/O line 1..16. |
| out | **AssignedChan* | Pointer to int value, where the assigned channel 1..8 is written to, but only if DIOtype= ThresholdSwitch (DIO_OUT_THRESHOLD_ANYVAL Bit set) or DIO_IN_TARE_SINGLE. |

**Returns:**

DIOtype:
DIO_IN_GENERALPURPOSE 0x04
DIO_IN_SYNC_SLAVE 0x02
DIO_IN_QEI_ANY 0x08
DIO_IN_TARE_SINGLE 0x10
DIO_IN_TARE_ALL 0x20
DIO_IN_RESET_MAXMIN 0x40
DIO_IN_RESET_DOUT 0x50
DIO_IN_TRIG_SEND_VAL 0x80
DIO_IN_TRIG_SEND_MAXVAL 0x100
DIO_IN_TRIG_SEND_MINVAL 0x200
DIO_IN_TRIG_SEND_AVGVAL 0x400
DIO_IN_TRIG_SEND_VAL_WHILE_HI 0x800
DIO_OUT_GENERALPURPOSE 0x1000
DIO_OUT_THRESHOLD_ANYVAL 0x10000
DIO_OUT_THRESHOLD_MAXVAL 0x14000
DIO_OUT_THRESHOLD_MINVAL 0x18000
DIO_OUT_SYNC_MASTER 0x20000
DIO_THRESHOLD_WINDOWCOMP_MASK 0x2000
DIO_INVERT_MASK 0x800000
or GSV_ERROR if function failed.
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

See annex for details on DIO-type.

OrdinalNo: 153

Device access: Yes, CmdNo: 0x5B

date: 18.06.2015

Applicable devices: GSV-8

### int CALLTYP GSV86getDoutInitLevel (int *ComNo*, int *DIOno*)

Reads the default level of the digital I/O line. This level is set on device boot up, eventually after a type change or if none of output-setting conditions are met (yet). Will be stored for all 16 I/O lines, but only meaningful for output types.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DIOno* | Number of digital IO line 0..16. 0: Get all 16 DO init levels, whereby Bit0 of return value = DIOno 1.. Bit15= DIOno 16 |


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

**Returns:**

DO init level: GSV_TRUE, if DO init level = high, GSV_OK, if level = low, or Bits<15:0> = all levels if DIOno=0 or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 160

Device access: Yes, CmdNo: 0x61

date: 01.07.2015

Applicable devices: GSV-8

### int CALLTYP GSV86getDoutThreshold (int *ComNo*, int *DIOno*, double * *ThresUp*, double * *ThresDown*)

Reads the two threshold values for digital output threshold switch. Will be stored for any digital I/O line, but only meaningful for digital output threshold types.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DIOno* | Number of digital IO line 1..16. |
| out | **ThresUp* | Pointer to double value, where upper threshold is written to |
| out | **ThresDown* | Pointer to double value, where lower threshold is written to |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 157 Device access: Yes, CmdNo: 0x5F

date: 21.06.2015

Applicable devices: GSV-8

### double CALLTYP GSV86getFrequency (int *ComNo*)

Reads the measuring data frequency, with that whole measuring data frames are transmitted, it permanent value transmission is enabled (default).

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Measuring data rate or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 48

Device access: Yes, CmdNo: 0x8A

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86getFTsensorActive (int *ComNo*)

Reads the enabled/disabled state for the Force/Torque multi-axis sensor.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

GSV_OK,if Six-axis sensor disabled, GSV_TRUE if enabled, or GSV_ERROR if function failed
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 134

Device access: Yes, CmdNo: 0x26

date: 30.04.2015

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86getFTsensorCalArrayInfo (int *ComNo*, int * *MaxNumSupp*, int * *ArrNumStored*)

Several parameter arrays can be stored for the Force/Torque multi-axis sensor, but only one is used (=active). This reads information about theses arrays.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | **MaxNumSupp* | Pointer to int value, where the maximum number of storable six-axis arrays are written |
| out | **ArrNumStored* | Pointer to int value, where the number of stored six-axis-arrays are written |

**Returns:**

Index of active six-axis calibration array (beginning with 0) or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 162

Device access: Yes, CmdNo: 0x54

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86getIDlevel (int *ComNo*)

Some write functions require the device password / User ID to be given. This function reads the state of the given user ID.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

ID level: 0: no password set
>=8: User password (=device password) set
or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 143

Device access: Yes, CmdNo: 0x1A

date: 16.04.2015

Applicable devices: GSV-8, GSV-6


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

**int CALLTYP GSV86getInterfaceIdentity (int *ComNo*,**
**int *StopPermTX*,**
**int * *NumOfIntfDescr*,**
**int * *ThisInterfNo*,**
**int * *WriteProtect*,**
**int * *MValDataType*,**
**int * *MValPermanenTX*,**
**int * *NumObjInMVframe*,**
**int * *ThisDeviceModel*,**
**int * *ThisProtocol*)**

Reads information about the devices communication interface(s).

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *StopPermTX* | If =1, The permanent measuring data transmission will be stopped. If =2, The permanent measuring data transmission will be started. |
| out | **NumOfIntfDescr* | Pointer to total number of communication interfaces and their descriptors |
| out | **ThisInterfNo* | Pointer to Index of this communicating interface in the descriptor array |
| out | **WriteProtection* | Pointer: =0: no write protection, =2: write protection enabled |
| out | **MValDataType* | Pointer to measuring value type: =1: DATATYP_INT16, =2: ~24, =3: DATATYP_FLOAT |
| out | **MValPermanenTX* | Pointer: =1 permanent measuring value transmission on, =0: no permanent val.tx |
| out | **NumObjInMVfra me* | Pointer to Number of objects in measuring value frame |
| out | **ThisDeviceModel* | Pointer to Model No of device attached. GSV-8: =8 |
| out | **ThisProtocol* | Pointer to General interface type. =1 for serial |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 7

Device access: Yes, CmdNo: 0x01

date: 30.09.2015

Applicable devices: GSV-8, GSV-6

**int CALLTYP GSV86getInTypeRange (int *ComNo*, int *Chan*, double * *Range*)**

Reads the analog input type and range.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Number of input channel, 1..10 |
| out | *Range* | If not NULL: pointer to one variable of double, where the input range (sensitivity) is written to. Physical unit of range depends on the Input Type: Bridge inputs (0..2): mV/V. Single ended (3): V Temperature (4..6): range is fixed to 1000°C; Counter (7): Maximum counter or frequency value, respectively |

**Returns:**

Input Type, defined as follows:
INTYP_BRIDGE_US875 0 Input-Type Bridge at Vexcitation=8.75V

INTYP_BRIDGE_US5 1 Input-Type Bridge at Vexcitation=5V
INTYP_BRIDGE_US25 2 Input-Type Bridge at Vexcitation=2.5V
INTYP_SE10 3 Input-Type single-ended +-10V
INTYP_PT1000 4 Input-Type Temperature-Sensor PT1000
INTYP_TEMP_K 5 Input-Type Temperature-Sensor Type-K
INTYP_TEMP_K_DT 6 Temperature-Sensor Type K, relative
INTYP_COUNTER 7 Counter/Frequency measuring input
or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 64

Device access: Yes, CmdNo: 0xA2

date: 11-18-2014

Applicable devices: GSV-8

### int CALLTYP GSV86getIsCmdAvailable (int *ComNo*, int *CmdUp*, int *CmdLo*)

Reads the information, whether one or several device commands are available.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *CmdUp* | Higher command number of command number range to check (may be equal to CmdLo, if only one command is to be checked) |
| in | *CmdLo* | Lower command number of command number range to check |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 146

Device access: Yes, CmdNo: 0x93

date: 16.04.2015 12:12

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86getLastDeviceError (int *ComNo*, int *Async*)

The device stores the last error, if a device command failed or if a precondition is subnormal. This returns this last error stored by the device.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Async* | =0: Last error of measuring application or command interface =1: Last error of protocol frames |

**Returns:**

Error code thrown by device, See Annex for details.

OrdinalNo: 127

Device access: Yes, CmdNo: 0x42

date: 23.02.2017

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86getLastErrorText (int *ComNo*, char * *ErrText*)

If the device returned an error on a device command, the DLL stored that. The DLL also catches error conditions that it detects itself. This function can be called after another function returned GSV_ERROR and it additionally retrieves an error describing text.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ErrText* | Pointer to ASCII coded 8-Bit char array of size 256 chars, where error text is written to. |

**Returns:**

One of the following:

1. Direct System Error Codes (http://msdn.microsoft.com/en-us/library/ms681381%28VS.85%29.aspx), retrieved by GetLastError(), which was called by previously failed function. Only if GSV86actEx/GSV86activateExtended failed before hardware were accessed.
2. System Error Codes as above ORed with OWN_ERR_MASK (0x20000000)
3. Error code for DLL-caught error as defined in Errorcodes.h. Range: 0x30000001..0x3000FFFF
4. Error code thrown by GSV-8 device, ORed with 0x38000000. Range: 0x38000001..0x380000FF. See GSV-8 manual for details.

OrdinalNo: 5

Device access: No

date: 27.04.2015

### int CALLTYP GSV86getLastProtocollError (int *ComNo*)

If the device returned an error on a device command, the dll stored that. So, this can an be called after another function returned GSV_ERROR and returns the error code.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

One of the following:

1. Direct System Error Codes (http://msdn.microsoft.com/en-us/library/ms681381%28VS.85%29.aspx), retrieved by GetLastError(), which was called by previously failed function. Only if GSV86actEx/GSV86activateExtended failed before hardware were accessed.
2. System Error Codes as above ORed with OWN_ERR_MASK (0x20000000)
3. Error code as defined above. Range: 0x30000001..0x3000FFFF
4. Error code thrown by GSV-8 device, ORed with 0x38000000. Range: 0x38000001..0x380000FF. See GSV-8 manual for details.

OrdinalNo: 6

Device access: No

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86getMaxMinValue (int *ComNo*, int *Chan*, double * *MaxValue*, double * *MinValue*)

If the corresponding mode flag is set (see GSV86getMode), the device determines maximum and minimum values. This function reads these values, independently of Txmode settings.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 0..8: Get maximum and Minimum value of specified channel<br>0 means: Get values of all measuring objects (8 channels, respective): |

| | | |
|---|---|---|
| | | If =0: *MaxValue and *MinValue must point to array of (at least) double[8] size |
| out | *MaxValue* | Pointer to single value of double type (Chan>0), where maximum measuring value will be written to. |
| out | *MinValue* | pointer to single value of double type (Chan>0), where minimum measuring value will be written to. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

Precondition: Maximum/Minimum Mode activated in device: GSV86getModeMaxMin must return GSV_TRUE

OrdinalNo: 142

Device access: Yes (Cmd 0x53)

date: 16.04.2015

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86getMode (int *ComNo*)

Reads device-mode flags.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:** Mode-Flags:

**GSV-8:**

Bit 0: SIX_AXIS_SENSOR_ACTIVE 0x01 Multi-axis sensor used
Bit 1: ANALOG_FILTER_AUTO 0x02 Analog input filter cutoff frequency switched automatically
Bit 2: MODE_MAXIMUM 0x04 Maximum and minimum values are determined
Bit 3: MODE_NOISECUT 0x08 Values around 0 (Noisecut-threshold dependent) are set to 0
Bit 4: Absolute measuring value used for maximum- and minimum checking
Bit 7: ALL_WRITES_BLOCKED All write functions are rejected
Bits <15:8>: Use TEDS for scaling of corresponding channel 8..1
Bit 16: Set Unit from TEDS (if channel-corresponding Bit in <15:8> set)
Bit 17: Set Input type from TEDS (if channel-corresponding Bit in <15:8> set)
Bit 18: Set Output scaling according to TEDS physical range (if channel-corresponding Bit in <15:8> set)
Bit 19: Set zero-point according to TEDS data (if channel-corresponding Bit in <15:8> set)
Bit 22: Auto-Zero active (see AutoZero functions)

**GSV-6:**


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

<2:0> Number of active channels
0b001: 1 channel
0b010: 2 channels
0b011: 3 channels
0b110: 6 channels
<3> Save-Tara
<4> User-Monitor
<5> Set Offset from TEDS
<6> Scale-Negate
<7> Peak-value output
<8> Peak-value-Reset and Tare
<9> Read TEDS at Boot
<10> FT-sensor calculation
or GSV_ERROR if function failed
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 32

Device access: Yes, CmdNo: 0x26

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86getModeAfilterAuto (int *ComNo*)

Reads the enabled/disabled setting of the automatic analog filter determination. If enabled, the device sets the analog input filter, depending on the device data frequency.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

GSV_OK,if Analog-filter-auto is disabled, GSV_TRUE if enabled, or
GSV_ERROR if function failed
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 136

Device access: Yes (Cmd. 0x26)

date: 30.04.2015

Applicable devices: GSV-8

### int CALLTYP GSV86getModeMaxMin (int *ComNo*)

Reads the enabled/disabled setting of the maximum and minimum values determination. If enabled, every measuring value is compared to the maximum and the minimum value stored, and updated accordingly.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

GSV_OK,if Maximum / Minimum mode disabled, GSV_TRUE if enabled, or GSV_ERROR if function failed

If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 145

Device access: Yes (Cmd. 0x26)

date: 19.05.2015

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86getModeNoiseCut (int *ComNo*)

Reads the enabled/disabled setting of the noise-cut filter. If enabled, the device sets values between the noise-cut threshold and its negation (i.e. around zero) to exactly zero.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

GSV_OK,if Noise-cut threshold disabled, GSV_TRUE if enabled, or
GSV_ERROR if function failed
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 138

Device access: Yes (Cmd. 0x26)

date: 30.04.2015

Applicable devices: GSV-8

### int CALLTYP GSV86getNoiseCutThreshold (int *ComNo*, int *Chan*, double * *Thres*)

If corresponding flag in Mode is set (see GSV86getMode() ), the device will set measuring values below this threshold (but above -1*threshold) to the zero value.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: Get noise cut threshold for specified channel |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 118

Device access: Yes, CmdNo: 0x94

date: 10-31-2014

Applicable devices: GSV-8

### int CALLTYP GSV86getSensorPlugged (int *ComNo*, int *Chan*, int * *BridgeSensor*, int * *TEDScapable*)

Reads information on sensors: Bridge sensor connected and TEDS (transducer electronic data sheet) connected and write-capable.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

| | | |
|---|---|---|
| in | *Chan* | Input channel No (0..8) |
| out | **BridgeSensor* | Flag value indicating, if a bridge sensor is connected (Flag=1) or not (Flag=0).<br>if Chan=0: Bits<7:0> correspond to bridge-sensor at input channels: Bit7: Input 8..bit0: Input 1<br>if Chan =1..8: Value=1 indicated a connected bridge sensor. =0: No bridge sensor connected. |
| out | **TEDScapable* | Indicates, if a 1-wire EEPROM is connected and if it is writable<br>If Chan=0: Bits<7:0> indicate present 1-wire EEPROM of all input channels 8..1 (Bit7: chan. 8...Bit0: chan 1)<br>Bits<15:8> indicate if the corresponding input channel is capable of writing to 1-wire TEDS EEPROM<br>Bit15: Input chan. 8 writable ... Bit0: Input chan. 1 writable<br>if Chan =1..8: Bit 0 indicates if TEDS is present at specified input channel<br>Bit 1 indicates if specified input channel is capable of writing TEDS |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

: The "1-wire EEPROM" flags indicate if a TEDS-capable memory is connected; it doesn't indicate if it contains valid TEDS data and if they are used for scaling. To determine the latter, use GSV86getTEDSactive instead.

OrdinalNo: 186

Device access: Yes, CmdNo: 0x45

date: 06.07.2016

Applicable devices: GSV-8

## int CALLTYP GSV86getSerialNo (int *ComNo*)

Reads the devices individual serial number.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Serial-No or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 70

Device access: Yes, CmdNo: 0x1F

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86getSoftwareConfiguration (int *ComNo*)

Reads information on the devices equipment.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Flags:
Bit 0: HAS_ADC Has analog-digital converter
Bit 1: HAS_ETHERCAT Has EtherCat field bus
Bit 2: HAS_LCD Has LC-Display
Bit 3: HAS_TEDS Has TEDS sensor reading capability
Bit 4: HAS_DIGI_IO Has digital inputs and outputs
Bit 5: HAS_ETH_TWOLEDS Ethercat with separated status-LED (DS housing)
Bit 6: HAS_ANALOG_OUT Has analog output(s)
Bit 7: HAS_SERIAL Has serial interface
Bit 8: HAS_FREQ_OUT Has frequency output
Bit 9: HAS_AIN_MCU Has Co processor for analog input
Bit 10: HAS_SIXAXIS Supports six-axis sensor
Bit 11: HAS_CANOPEN Has CANopen fieldbus
or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 122

Device access: Yes, CmdNo: 0x2A

date: 10-31-2014

Applicable devices: GSV-8

## int CALLTYP GSV86getTEDSactive (int *ComNo*, int *Chan*)

Reads information whether parametrization was done by the data content of a TEDS (transducer electronic data sheet) sensor.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | GSV-8: Input channel No (0..8) if =0: TEDS active flags in Bity7:0> for input channels 8..1<br>GSV-6: Not used (TEDS available for channel 1 only) |

**Returns:**

If Chan=0: TEDS active flags in Bits<7:0> for input channels 8..1
if Chan= 1..8: TEDS active flag in Bit 0, whereby Bit=1: TEDS are actually used for scaling.
Bit=0: no TEDS data used for scaling, scaling used from GSV-8 memory

**Remarks:**

Preconditions for TEDS data used:

- Mode-Flags<15:8> of corresponding channel =1 (see GSV86getMode)
- TEDS with valid and known template connected
- Suitable input-Type set (see GSV86getInTypeRange)

OrdinalNo: 188

Device access: Yes, CmdNo: 0x68

date: 06.07.2016 15:29

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86getTXmode (int *ComNo*, int *Index*)

Reads information about the measuring data transmission mode.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

| | | |
|---|---|---|
| in | *Index* | **=0: Read Txmode-flags:** Return value defined as follows:<br>Bit0: =1: Measuring value transmission is temporarily stopped (by GSV86stopTX) =0: Permanent measuring value transmission active<br>Bit1: =1: Measuring value transmission is stopped (stored in non-volatile memory) =0: By parametrization (non-volatile memory), permanent value transmission is active Remark: Get this information simpler by calling GSV86isValTXpermanent<br>Bit 6: =1: GSV-8 only: Serial Port (Ethernet) has no write access =0: Serial Port (Ethernet) may have write access (depending on other settings)<br>Bit 7: =1: GSV-8 only: USB Port has no write access =0: USB Port may have write access (depending on other settings) Remark: Get this information simpler and reliably by calling GSV86getWriteAccess.<br>**=1: Read measuring value type:** Return value defined as follows:<br>1: Measuring value is a 16-Bit integer, unscaled raw value<br>2: Measuring value is a 24-Bit integer, unscaled raw value (GSV-8 only)<br>3: Measuring value is a IEEE754 float, 32 bit size, scaled in physical units<br>**=2: Read maximum number of channel objects** Identify the maxum number of values in the measuring value frame.<br>**Remarks**: - The data interpretation of 16-Bit data type differs: GSV-8: Binary offset format (see manual) GSV-8 Signed Int16.<br>- The value type can also be determined with GSV86getValObjectInfo **=2: Read input channel numbering information:**<br>GSV-8: Return value defined as follows: Bits<7:0>: lowest input channel number (constant =1 for GSV-8 by now) Bits<15:8>: highest input channel number (constant =8 for GSV-8 by now)<br>GSV-6: Maximum number of channels (=6) |
| | | |

**Returns:**

Informational value as defined above, or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 22

Device access: Yes, CmdNo: 0x80

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86getUnitNo (int *ComNo*, int *Chan*)

Read number of enumerated physical unit.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..10: Channel number of unit number to read |

**Returns:**

Unit-number. For assignment of codes 0..UNIT_NO_MAX to unit itself, see manual / annex.
special codes: 0x000000FF: User definable unit text 1 active
0x000000FE: User definable unit text 2 active
or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 80

Device access: Yes, CmdNo: 0x0F

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86getUnitText (int *ComNo*, int *Chan*, int *Code*, char * *UnitText*)

Read physical unit name.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: Channel number of active unit text to read |
| in | *code* | Control value, consisting of unit type (Bits <7:0> and code page value (Bits <23:8>)<br>**Bits <23:8>: Code page:** Only used if unit set by UnitNo (Bits<7:0>=0). Supported values:<br>ANSI_CODEPAGE =0x0000: ANSI 8-bit<br>ASCII_ONLY 0x0001: Strictly 7-Bit ASCII only (micro sign ->'u', degree sign left away, per mille->0/00)<br>DOS_CODEPAGE_437 0x01B5 (437d): Windows/DOS codepage 437<br>WIN_CODEPAGE_1252 0x04E4 (1252d): Windows codepage 1252<br>**Bits<7:0>: Unit type to read:**<br>ACTIVE_UNIT_ANY =0: Read active unit (as set by unit-no, see GSV86getUnitNo)<br>USER_UNIT_1 =1: Read user-defined unit string number 1 (Chan ignored)<br>USER_UNIT_2 =2: Read user-defined unit string number 2 (Chan ignored) |
| out | *UnitText* | Pointer to char array: Unit-Text to read, coded as defined by *code*. Must be at least 8 chars in length |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 84

Device access: Yes, CmdNo: 0x0F and eventually 0x11

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86getValMapping (int *ComNo*, int *Index*)

Reads information about measuring values contained in the transmitted value frame.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Index* | **0:** number of mapped objects.<br>**1..**NumberOfMappedObjects: Get Info of corresponding Object in Value-Frame, whereby 1: first Object in device value frame, which is at Object Index 0 of GSVread |

**Returns:**

If Index=0: number of mapped objects (1..16) or =0: Mapping not present or not supported by device else if Index= 1..NumberOfMappedObjects:
ObjMapping: See description for GSV86getValObjectInfo()
or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

OrdinalNo: 38

Device access: Yes, CmdNo: 0x49

date: 10-31-2014

Applicable devices: GSV-8

### int CALLTYP GSV86getValObjectInfo (int *ComNo*, double * *ScaleFactors*, unsigned long * *ObjMapping*, int * *DataType*)

Get info about the measuring values (read by GSV86read/GSV86readMultiple) interpretation.

Typically used once before measuring loop has started and after NormFactor(s) or Mode-States had changed.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *ScaleFactors* | Pointer to array of double with size = 16.<br>If pointer-value is not NULL, the functions writes Factors into indices 0.. (NumberOfMappedObjects-1), with which the corresponding measuring value-objects should be multiplied in order to get values that are scaled in physical units. |
| out | *ObjMapping* | Pointer to array of concatenated enumerations with size = 16.<br>If pointer-value is not NULL, the functions writes into indices 0.. (NumberOfMappedObjects-1) the mapping information of the corresponding measuring-value-objects.<br>This Mapping Info is a quasi-struct and defined as follows:<br>Low-byte**, Bits<7:0>:** Amplifiers input channel-Number of corresponding value-object. Range is 1 to 8.<br>**Bits<15:8>: Value-Type** of corresponding value-object, defined as follows:<br>NORMAL=0x00: The value-object is an actual value<br>MAX_VAL=0x01: The value-object is a maximum value<br>MIN_VAL=0x02: The value-object is a minimum value<br>**Bits<23:16>:**<br>Physical type corresponding value-object, defined as follows:<br>0x00: No physical type defined<br>0x01: Force in X direction (mainly with Six-axis sensors)<br>0x02: Force in Y direction (mainly with Six-axis sensors)<br>0x03: Force in Z direction (mainly with Six-axis sensors)<br>0x04: Torque in X direction (mainly with Six-axis sensors)<br>0x05: Torque in Y direction (mainly with Six-axis sensors)<br>0x06: Torque in Z direction (mainly with Six-axis sensors)<br>0x10: Raw value of six-axis sensor (before calculation)<br>0x20: Temperature<br>0x28: QEI / Counter value<br>0x38: Quadrature-Encoder-Speed / Frequency |
| out | *DataType* | Pointer to one 32-Bit enumeration value.<br>If pointer-value is not NULL, the function writes one of the following codes for the data type of the measuring values in the value frame sent by the GSV-8 (the values in the frame have all the same data type):<br>0x01: Data type sent by device is 16-Bit integer value in binary offset format (raw value)<br>0x02: Data type sent by device is 24-Bit integer value in binary offset format (raw value)<br>0x03: Data type sent by device is 32-Bit float value (IEEE754), scaled in physical units |

**Returns:**

NumberOfMappedObjects: Number of value-objects in the measuring frame, Range: 1 to 16. Or GSV_ERROR if function failed. If =GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 36

Device access: Yes, CmdNo: 0x26, 0x80, eventually 0x14

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86getValueError (int *ComNo*, int *Ix*, int * *ErrInfo*, double * *ErrTime*)

Reads devices measuring application faults that are stored by device.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Ix* | **=0:** (GSV-8) Get number of Error events stored in Memory as return value.<br>If >=1: First Error (number) is a recent one, stored in memory, accessible by index 1.<br>>=2: (GSV-8 only) Error(s) stored in memory, accessible from index 2 on, up to <Number of Errors> + 1.<br>**=1:** (GSV-8 only) Get recent Error: Writes error flags to *ErrInfo and device hours, when error occurred to *ErrTime.<br>**= 2..<Number of errors +1>:** (GSV-8 only) get Error(s) stored in memory (see annex) |
| out | **ErrInfo* | Pointer to Int value, where Error Flags are written to. See annex for description; GSV-8 only |
| out | **ErrTime* | Pointer to Double value, where device hours, when error occurred is written to . (GSV-8 only) |

**Returns (GSV-8):**

Number of errors stored, if Ix =0, otherwise:
Error type:
VALERR_TYPE_SATURATED 1 Measuring value saturated
VALERR_TYPE_MAX_EXCEED 2 For Multi-axis sensors: Maximum exceedance
VALERR_TYPE_SENSOR_BROKEN 3 Sensor or sensor cable broken
HWERR_TYPE_ANA_OUT 4 Analog output connection error (e.g. current output open)
HWERR_TYPE_DIO 5 Digital output error (shorted)
or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Returns (GSV-6):**

Bits<15:8>: Saturation of input value of corresponding channel: Bit13: Ch.6 ... Bit 8: Ch.1

Bits <7:0>: Maximum value exceedance with Six-Axis sensors: Bit13: Ch.6 (Mz) ... Bit 8: Ch.1 (Fx)

or GSV_ERROR as described above.

**Note:**

Refer to Manual Appendix for Error Flag coding

OrdinalNo: 126

Device access: Yes, CmdNo: 0x43

date: 09.04.2015

Applicable devices: GSV-8, GSV-6 (limited to index 0 and return value bytes <1:0>)


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

### int CALLTYP GSV86getWriteAccess (int *ComNo*)

Reads information, whether the interface used has permission to alter settings of the device. Write-access can be blocked by user-setting or because another active interface has write-access.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

GSV_OK, if this interface doesn't have write access
GSV_TRUE, if this interface has write access
or GSV_ERROR if function failed
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 60

Device access: Yes, CmdNo: 0x26

date: 11-19-2014

Applicable devices: GSV-8

### int CALLTYP GSV86isValTXpermanent (int *ComNo*)

Reads information about permanent measuring value frame transmission.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

GSV_OK, if permanent value transmission is actually stopped
GSV_TRUE, if permanent value transmission is actually active
or GSV_ERROR if function failed
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 28

Device access: Yes, CmdNo: 0x80

date: 11-19-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86loadSettings (int *ComNo*, int *DataSetNo*)

Restores alternative device parametrization previously stored by user.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DataSetNo* | =0: Load settings from previous power-on session =1: Load factory default settings =2..6: Load settings, which were previously stored by user |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

Load all device configuration parameters, previously stored by user (DataSetNo 2..6) Except the following settings, which are NOT part of the user parameter array:

Interface Communication settings
Measuring value Type
Digital FIR/IIR filter configuration itself (but Filter On/Off state is part of user parameter)
Six-Axis-Sensor configuration itself (but en-/disabled state and active FT-Array-No is part of user parameter)

OrdinalNo: 76
Device access: Yes, CmdNo: 0x09
date: 10-31-2014
Applicable devices: GSV-8, GSV-6 (limited to *DataSetNo* 0..1)

### int CALLTYP GSV86read (int *ComNo*, int *Chan*, double * *out*)

Reads one measuring value of one input channel from the dlls measuring value buffer.
The interpretation of *out* depends on the measuring value type (see GSV86getValObjectInfo):
Float: The value is readily scaled to physical representation (provided the parametrization with UserScale or Six-axis sensor is correct).
Int16 / Int24: The range of *out* lies between -1.0 and 1.0 (with an 5% overhead it's from -1.05 to 1.05 total).
The value 1.0 represents the positive full-scale input value, depending on Input type: (see GSV86setInType).

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Number of Object (channel, respectively) to read. Range: 1..NumMappedObjects |
| out | *out* | Pointer to one double value, were the measuring value is written to |

**Returns:**

GSV_OK,if no value was read (buffer empty), or
GSV_TRUE if a value was read, or
GSV_ERROR if an error occurred If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 12
Device access: No
date: 10-31-2014

### int CALLTYP GSV86readAllInterfSettings (int *ComNo*, int *IntNo*, int * *IntfEnums*, int * *Dtypes*, int * *Data*, int * *BdList*, int * *BdNum*)

Reads information about existent communication / fieldbus interfaces.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *IntfNo* | Interface number: 0..Number of interfaces-1 |
| out | *IntfEnums* | Always 2 values written: IntfEnums[0]: Physical type enum, IntfEnums[1]: Application type enum |
| out | *Dtypes* | Array with all extended settings written, except baud rates. =0 basic flag value<br>>0: Type enumerator for Data meaning |


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

| | | |
|---|---|---|
| out | *Data* | Data content array. Interpretation according to Dtypes at same index. |
| out | *BdList* | Array with all baud rates settings written [Bits/s]. bdList[0]: actual baud rate set. |
| out | *BdNum* | value that holds number of baud rates written to bdList. bdNum-1 =Number of available baud rates |

**Note:**

Array size for Dtypes, dat and bdList won't ever exceed 256 Bytes. Just allocate 256 values.

**Returns:**

Number of values written to dat and Dtypes or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 174

Device access: Yes, CmdNo: 0x01, 0x7B

date: 24.01.2016

Applicable devices: GSV-8

### int CALLTYP GSV86readAnalogFilterCutOff (int *ComNo*, double * *CutOffFreq*)

Reads -3dB cut-off frequency of the analog input filter.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *CutOffFreq** | pointer to double value where cutoff frequency of analog input filter is written to |

**Returns:**

GSV_TRUE, if cutoff frequency is set by device automatically, according to the data rate (see device manual). GSV_OK, if automatic filter setting is disabled or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 128

Device access: Yes, CmdNo: 0x90

date: 16.04.2015

Applicable devices: GSV-8

### int CALLTYP GSV86readAnalogOutOffset (int *ComNo*, int *Chan*, double * *Offset*)

Reads the offset value of the analog output (representing zero at analog input).

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: Channel number of offset value to read |
| out | *Offset* * | Pointer, where function will write the Offset value in percentage of positive full-scale value. See above. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 102

Device access: Yes, CmdNo: 0x04

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86readAnalogOutScale (int *ComNo*, int *Chan*, double * *Scale*)

Reads the user-defined scaling value, with which the analog output can be scaled to get a specific output value a specific input (GSV-8). With GSV-6, this value is an input scaling factor, scaling the input range down to a desired input sensitivity.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: Channel number of scale value to read |
| out | *Scale* * | Pointer, where function will write the scaling value.<br>GSV-8: This is a factor representing the deviation of the output range based on the type. A scaling value of 1.0 will result in the positive full-scale voltage or current at the output if the Offset is =0 and the measuring value is either the input full range value (based on the input type) or the sensor maximum value (with six-axis sensors). |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 106

Device access: Yes, CmdNo: 0x06

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86readBasicInterfSettings (int *ComNo*, int *ActIntf*, int * *PhysEnums*, int * *ApplEnums*, int * *Flags*)

Reads some fundamental settings (e.g. type) of existent communication / fieldbus interfaces.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ActIntf* | =0 get List of all Interfaces, =1: Get Actual Interface settings |
| out | *PhysEnums** | ActIntf=1: <IntfNo> values written, else 1 value: Physical type enum, Index= Interface-No, see annex |
| out | *ApplEnums* * | IntfNo values written, else 1 value: Application Protocol Enum, see annex |
| out | *Flags* | IntfNo values written, else 1 value, see annex |

**Returns:**

If ActIntf=1: Number of actual Interface, e.g. interface with that the request was done.
if ActIntf=0: Number of Interfaces present =Number of values written in Flags, PhysEnums and ApplEnums Arrays or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 173

Device access: Yes, CmdNo: 0x01, 0x7B

date: 24.01.2016

Applicable devices: GSV-8

### int CALLTYP GSV86readCANsettings (int *ComNo*, int *Index*, int * *setting*)

Reads information on CAN interface settings.


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Index* | **=0:** Command ID GSV86-CAN protocol<br>**=1:** Command-Answer ID GSV86-CAN protocol<br>**=2:** Value-Frame ID GSV86-CAN protocol<br>**=3:** Multicast ID GSV86-CAN protocol<br>**=4:** CAN-Baudrate in Bits/s (both CAN protocols)<br>**=5:** Flags: Bit 1: CAN appl. layer protocol: =0: GSV86-CAN protocol. =1: CANopen protocol<br>Bit 0: =0: CAN interface is switched on. =1: CAN interface is switched off<br>**=6:** CANopen NodeID (range: 1..0x7F) |
| out | **setting.* | Pointer to value, where setting (according to Index) is written to |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 167

Device access: Yes, CmdNo: 0x8C

date: 24.11.2015

Applicable devices: GSV-8, GSV-6

### double CALLTYP GSV86readDeviceHours (int *ComNo*, int *Index*)

Reads the devices operating hours.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Index* | =0: Get absolute Device working hours (not resettable)<br>=1: Get relative (settable) device working hours |

**Returns:**

: Hours, which the device was switched on or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 116

Device access: Yes, CmdNo: 0x56

date: 26-02-2015

Applicable devices: GSV-8, GSV-6 with RTC (GSV-6BT)

### int CALLTYP GSV86readFormattedTEDSList (int *ComNo*, int *Chan*, const char * *TEDSfilePath*, char * *ListOut*, int *ListSize*, int *Code*, char * *ExtListOut*)

Writes a text list containing all data of a connected TEDS (transducer electronic data sheet) sensor.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Input channel No (1..8) with TEDS sensor connected |
| in | **TEDSfilePath* | Full Path to TEDSdictionary.ini file |

| | | |
|---|---|---|
| out | **ListOut* | String list. One line per entry, ending with LF CR. Format:<br>1 2 3<br>Name Value Unit |
| in | *ListSize* | Size of ListOut String in Bytes. Minimum recommended: 16384 |
| in | *Code* | Bits<7:0>: Flags: see #TEDSLISTFLG_ above Bits<23:0> of Code parameter. Can be ORed with Bit<7:0> constants<br>ANSI_CODEPAGE 0x00000000 ANSI 8-Bit coded<br>ASCII_ONLY 0x00000100 Use ASCII 7-Bit only<br>DOS_CODEPAGE_437 0x0001B500 DOS/Windows Codepage 437<br>WIN_CODEPAGE_1252 0x0004E400 Windows Codepage 1252 |
| out | **ExtListOut* | String list with additional information. One line per entry, ending with LF CR. Format:<br>1 2 3 4<br>Extended Text (Enum-)Value Property-Name Property-ID |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:** The file **TEDSdefinitions.ini** must be present in the callers process directory. That file must support the template used by the connected TEDS sensor.

OrdinalNo: 182

Device access: Yes, CmdNo: 0x64

Applicable devices: GSV-8

**int CALLTYP GSV86readFTsensorCalArray (int *ComNo*,**
**int *ArrNo*,**
**char * *SensorSerNo*,**
**double * *MatrixNorm*,**
**double * *InSens*,**
**double * *Matrix*,**
**double * *Offsets*,**
**double * *MaxVals*,**
**double * *Zvals*)**

Reads Force-torque (multi-axis) sensors parameters. For single-matrix (standard) sensors, these are all parameters. Additinally, with GSV-8, GSV86readFTsensorCalArrExt may be called for sensors with matrix plus.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ArrNo* | 0x80: TEDS storage (coming versions), Bits <6:0>: Calibrations structs index |
| out | *SensorSerNo* | pointer to String of 8 number digits: Sensor serial number will be written here |
| out | *MatrixNorm* | ptr to single double value: Scaling of calibration matrix will be written here |
| out | *InSens* | ptr to single double value: input sensitivity of origin of calibration matrix ill be written here |
| out | *Matrix* | ptr to array of 36 double values: Calibration matrix will be written, index order: first row 0..5, 2nd row 6..11... |
| out | *Offsets* | ptr to array of 3 double values: mechanical offsets of sensor |

| | | |
|---|---|---|
| | | installation in meters. 0..2: Ox,Oy,Oz |
| out | *MaxVals* | ptr to array of 6 double values: maximum values for the 6 outputs. Order: 0..5: Fx,Fy,Fz,Mx,My,Mz |
| out | *Zvals* | ptr to array of 6 double values: No-load sensor deviations of the 6 inputs. Order: 0..5: Fx,Fy,Fz,Mx,My,Mz (GSV-8 only) |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 42

Device access: Yes, CmdNo: 0x47

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86readFTsensorCalValue (int *ComNo*, int *typ*, int *ix*, double * *val*)

Reads a Force-torque (multi-axis) sensors parameter

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *typ* | One of theses:<br>SENSORCAL_TYP_MATRIX_NORM 1 Scaling of calibration matrix. ix must be =0<br>SENSORCAL_TYP_MATRIX 2 calibration matrix. ix: 0..35<br>SENSORCAL_TYP_OFFSET 3 mechanical offsets of sensor installation. ix: 0:X,1:Y,2:Z, in meters<br>SENSORCAL_TYP_MAXVAL 4 maximum values for the 6 outputs. ix:0..5:Fx,Fy,Fz,Mx,My,Mz<br>SENSORCAL_TYP_INSENS 5 input sensitivity of origin if calibration matrix. ix must be =0<br>SENSORCAL_TYP_ZEROVAL 6 No-load sensor deviations of the 6 inputs. ix: 0..5 (GSV-8 only)<br>7: Sensortype (GSV-8, FWver >= 1.39):<br>Type=0: 6-Axis with Standard Matrix, 1st order only<br>Type=1: 6-Axis with matrix 1st and 2nd Order<br>Type=2: 3/4-Axis sensor (reserved)<br>8: Matrix B 2nd order values ix: 0..35<br>9: Channel Indices of first input factors ix: 0..5<br>10: Channel Indices of 2nd input factors ix: 0..5 |
| in | *ix* | Index, if array value, see typ. |
| out | *val* | Pointer to value, where value is written to |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:** If several calibration data structs are stored, the one that should be read must be set with GSV86setFTarrayToRead before (GSV-8 only).

OrdinalNo: 40

Device access: Yes, CmdNo: 0x47

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86readFTsensorSerNo (int *ComNo*, int *ArrNo*, char * *SensorSerNo*)

Reads the Force-torque (multi-axis) sensors serial number

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ArrNo* | Bits <6:0>: Calibrations structs index for storing several sensors/calibration data sets |
| out | *SensorSerNo* | pointer to String of 8 number digits: Sensor serial number will be written here |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 46

Device access: Yes, CmdNo: 0x47

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86readHWversion (int *ComNo*, int * *MainHW*, int * *ExtHW*)

Reads the hardware version number of the device

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | **MainHW* | Version of Devices main-board. Pointer required |
| out | **ExtHW* | Version of peripheral board(s), if relevant. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 179 Device access: Yes, CmdNo: 0x36

date: 05.07.2016

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86readInterfaceSetting (int *ComNo*, int *Ix*, int * *Next*, unsigned long * *Data*, int * *ApplEnum*, int * *Writable*)

The settings of the available communication interfaces are stored as a linked list. Reads an entry of that.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Ix* | index in linked list of interface descriptors/settings.<br>if Ix = 0..(Number of interfaces - 1) =Interface Number: Basic settings, ApplEnum set |
| out | *Next* | Next index in linked list of interface descriptors/settings of the same interface.<br>if =0: End of list |
| out | *Data* | Data read. if Ix= Interface Number: Flag value. Else: Interpretation according to return value, see annex |
| out | *ApplEnum* | If Index < (Number of interfaces): Enumerator of application |

| | | |
|---|---|---|
| | | interface, see annex. Otherwise: =0 |
| out | *Writable* | Function writes 1, if value is writable (with GSV86writeInterfaceSetting), otherwise 0 |

**Returns:**

If Ix= Interface Number: Enumerator of physical interface, see annex
otherwise: Enumerator of *Data* meaning, see annex

or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 172

Device access: Yes, CmdNo: 0x01, 0x7B

date: 24.01.2016

Applicable devices: GSV-8, eventually GSV-6

### int CALLTYP GSV86readMultiple (int *ComNo*, int *Chan*, double * *out*, int *count*, int * *valsread*, int * *ErrFlags*)

Reads several measuring values of one or all input channel(s).

The interpretation of the values in *out* depends on the measuring value type (see GSV86getValObjectInfo):

Float: The value is readily scaled to physical representation (provided the parametrization with UserScale or Six-axis sensor is correct).

Int16 / Int24: The range of *out* lies between -1.0 and 1.0 (with an 5% overhead it's from -1.05 to 1.05 total).

The value 1.0 represents the positive full-scale input value, depending on the Input type of the particular input channel (see GSV86setInType).

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | =0: Read all existent (channel-)objects<br>=1..NumMappedObjects: Read values of specified object |
| out | **out* | Pointer to (one-dimensional) array of double, where measuring values are written to (if return-value =GSV_TRUE).<br>Important: Array size must be greater or equal <count>! |
| in | *count* | Total maximum number of values to read.<br>If Chan=0, count must be dividable by NumMappedObjects. |
| out | **valsread* | Pointer to int value, where the actual number of values written into *out is stored. |
| out | **ErrFlags* | Pointer to int value, where measuring-value-related error flags given by device are stored. May be set to NULL, if not used.<br>Bit 0: Value saturation<br>Bit 1: Sensor range exceedance.<br>See annex |

**Returns:**

Simple error code: GSV_OK, if (at least one) buffer is empty and no values were read, or GSV_TRUE if value(s) were read, or
GSV_ERROR if an (internal) error occurred.
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

Value-related error flags set by device have no impact on the return value (i.e. *ErrFlags !=0 : return GSV_TRUE)

Reads whole array of measuring values in dll-value-buffer(s), if returned GSV_TRUE.

Reads up to <count> values and writes number of values, which were really read, into <valsread>. <count> must be >0. If resulting <valsread> is smaller than <count>, the whole buffer (the smallest, if Chan=0) was read.

If Chan=0: Reads whole Array of all channel-objects into the array of double-type, where <out> points to. <count> must be dividable by <NumMappedObjects> (determine NumMappedObjects with GSV86getValObjectInfo).

Assuming Chan=0 and NumMappedObjects=8, <out> is sorted as follows:

Obj1 in n*8, Obj2 in (n*8)+1, Obj3 in (n*8)+2, and so on, Obj8 in (n*8)+7; with n= {0...(<valsread> / 8)-1}

Example: Assumed, Chan=0, count >=16 and NumMappedObjects=8, and <valsread> results =16, the content of <out> is, beginning with index 0:

oldObj1,oldObj2,...,oldObj8,newObj1,newObj2,...,newObj8

If Chan={1..NumMappedObjects}: Reads whole Array of specified object.

Oldest value is at index 0 (base) of <out>-array, newest at <valsread>-1.

OrdinalNo: 14

Device access: No

date: 10-31-2014

### int CALLTYP GSV86readTEDSentry (int *ComNo*, int *Chan*, int *TemplID*, int *PropID*, int * *Next*, int *No*, unsigned long * *Udata*, double * *DblData*, int * *Flags*)

Reads an entry of TEDS (transducer electronic data sheet) data linked list. Entrys are identified by their property-IDs (see PropName entries in file TEDSdefinitions.ini. This file must not necessarily be present)

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Input channel No (1..8) with TEDS sensor connected |
| in | *TemplID* | Template ID. BasicTEDS (incl. ID main template): =0 |
| in | *PropID* | Property-ID of entry. 124 IDs exist so far. Property-ID =0: Get first valid entry, see annex |
| out | **Next* | Property-ID of next entry in list. If=0: last entry |
| in | *No* | Array index for same Property-ID beginning with 0 (reserved). Else =0 |
| out | **Udata* | Data of unsigned long type, if Flags written =0 |
| out | **DblData* | Data of double type, if Flags written =ANSW_IS_FLT =1 |
| out | **Flags* | 0..7F (Bit7 =0): Read fully successful. Bits<6:0>: Flag values:<br>TEDS_ANSW_IS_FLT 1 Data type of answer is float. Data written to *DblData.<br>TEDS_IS_PACKED_CHR5 2 Data type of answer is packed 5-Byte Char (Defined in IEEE1451.4 7.4.5.2.5)<br>TEDS_IS_DATE_DAYS 4 Data type of answer is date in days (Defined in IEEE1451.4 7.4.5.2.1)<br>0x80..FF (Bit7 =1): Special Error-Code, no value written<br>TEDS_ENTRY_NOT_EXIST 0xFF Entry-Request / PropID doesn't exist in the Template<br>TEDS_ENTRY_NOT_SET 0xFE Entry exists, but is flagged as "don't care", i.e. all Bits=1<br>TEDS_ENTRY_INVALID 0xFD Entry invalid, e.g NaN with Float |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 180

Device access: Yes, CmdNo: 0x64

Applicable devices: GSV-8

### int CALLTYP GSV86readTEDSrawData (int *ComNo*, int *Chan*, unsigned char * *DataOut*, int *NumBytes*, int *StartByteAdr*)

Reads binary raw data from TEDS sensor

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Input channel No (1..8) with TEDS sensor connected |
| out | **DataOut* | Array where data is written to. Size must be =NumBytes |
| in | *NumBytes* | Number of Bytes to read |
| in | *StartByteAdr* | Byte address of 1-wire EEPROM. Must be dividable by 4. TEDS-checksum not included. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: DataOut doesn't contain the Check-sum Byte(s); the function itself will check that.

OrdinalNo: 183

Device access: Yes, CmdNo: 0x65

date: 05.07.2016

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86read1wire (int *ComNo*, int *Chan*, unsigned char * *DataOut*, int *NumBytes*, int *StartByteAdr*)

Reads binary raw data from 1-wire memory device

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Input channel No (1..8) with 1-wire device connected |
| out | **DataOut* | Array where data is written to. Size must be =NumBytes |
| in | *NumBytes* | Number of Bytes to read |
| in | *StartByteAdr* | Byte address of 1-wire EEPROM. Must be dividable by 4. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: From main FW-version 1.53. Co-Processor Version 2. Fails otherwise.

OrdinalNo: 244

Device access: Yes, CmdNo: 0x65

date: 17/01/2020

Applicable devices: GSV-8

### int CALLTYP GSV86readUserOffset (int *ComNo*, int *Chan*, double * *Offset*)

If the measuring value data type is set to Float (see GSV86getValObjectInfo), the device can add a user-defined offset to

every measuring value. This function reads it.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..10: Read offset of specified channel |
| out | **Offset* | Pointer to double value, where the user offset is written to, if returned GSV_OK. This value is added by the device to every measuring value, if the device's measuring value data type is DATATYP_FLOAT |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 72

Device access: Yes, CmdNo: 0x9A

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86readUserScale (int *ComNo*, int *Chan*, double * *Norm*)

The device can be parametrized to have measuring values scaled in physical units. This reads that.

If the measuring value data type is set to one of the integer types (see GSV86getValObjectInfo), values retrieved by GSV86read or GSV86readMultiple should be multiplied with the User Scale by the caller in order to get physically scaled values.

With data type set to Float, the device itself multiplies raw values with the User Scale (except with six-axis and temperature sensor, where the user scale is meaningless).

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Number of input channel to get scaling factor. Range: 1..10 |
| out | **Norm* | Pointer to one variable of type double, where scaling factor is written to. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 28

Device access: Yes, CmdNo: 0x14

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86readZeroValue (int *ComNo*, int *Chan*)

The device always adds an offset to raw measuring values. That one is altered by a setZero routine. This function reads that raw offset value.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: channel to read zero value |

**Returns:**

Offset-zero raw value or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 141


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

Device access: Yes, CmdNo: 0x02

date: 16.04.2015

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86received (int *ComNo*, int *Chan*)

Determine, whether there are measuring values in the dlls reading thread buffers and how many.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | =0: get lowest buffer filling<br>=1..NumMappedObjects: get buffer filling for specified object<br>=NumMappedObjects+1: get highest buffer filling<br>Remark: Different buffer fillings can only occur, if GSV86read or GSV86readMultiple with parameter Chan >0 is used.<br>If GSV86readMultiple with parameter Chan=0 is used only, all buffers have the same filling. |

**Returns:**

Number of values in specified buffer, or GSV_ERROR if an error occurred. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 10

Device access: No

date: 10-31-2014

### int CALLTYP GSV86release (int *ComNo*)

Releases a communication with the device at specified COM-port and closes their connection and the port.

Should be called at program end (i.e. dll callers termination; must be called before re-opening a GSV-8 at the same COMport

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 20

Device access: GSV-6: No. GSV-8: Yes, Cmd. 0x7A

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86resetErrorState (int *ComNo*)

Resets the last errors in the device.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

Resets volatile (temporary) error state. Does not erase fault memory. OrdinalNo: 140

Device access: Yes (Cmd. 0x00)

date: 09.04.2015

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86setAnalogOutType (int *ComNo*, int *Chan*, int *Type*, int *Mode*)

Writes the type of the analog output.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 0..8: Channel number of analog output to change type and mode.<br>A value of 0 means that all outputs are set to the same type/mode. |
| in | *Type* | Analog output type enum as follows:<br>AOUT_TYPE_0_10V 0 0..10V<br>AOUT_TYPE_10_10V 1 -10..10V<br>AOUT_TYPE_0_5V 2 0..5V<br>AOUT_TYPE_5_5V 3 -5..5V<br>AOUT_TYPE_4_20A 4 4..20mA<br>AOUT_TYPE_0_20A 6 0..20mA |
| in | *Mode* | Flag value as follows:<br>Bit 0: AOUT_MODE_DIRECT =1 Output is on, but does NOT react on measuring values, but is direcly settable with GSV86writeAoutDirect<br>AOUT_ACTIVE_M_VALUES =0 Output is on and follows the corresponding measuring value input.<br>Bit 1: =1: AOUT_MODE_OFF 2 Output is off and high-impedance.<br>Bit 2: =1: Alternate Input (Out-Channels 7 or 8 only: linked to Counter/Frequency input) |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 114

Device access: Yes, CmdNo: 0x0E

date: 10-31-2014

Applicable devices: GSV-8

### int CALLTYP GSV86setCANonOff (int *ComNo*, int *OnOff*)

Enables/disables the CAN fieldbus.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *OnOff* | =0: Switch CAN/CANopen off. =1: Switch CAN/CANopen on |

**Returns:**

: Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 170

Device access: Yes, CmdNo: 0x8C, 0x8D

date: 02.12.2015


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

Applicable devices: GSV-8, GSV-6 (from FW-ver 3.25)

### int CALLTYP GSV86setCANsettings (int *ComNo*, int *Index*, int *setting*)

Writes settings of the CAN fieldbus.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Index* | **=0:** Command ID GSV6-CAN protocol<br>**=1:** Command-Answer ID GSV6-CAN protocol<br>**=2:** Value-Frame ID GSV6-CAN protocol<br>**=3:** Multicast ID GSV6-CAN protocol<br>**=4:** CAN-Baudrate in Bits/s (both CAN protocols)<br>**=5:** Flags. Bit 1: CAN appl. layer protocol: =0: GSV6-CAN protocol (read-only).<br>=1: CANopen Bit 0: =0: CAN interface is switched on. =1: CAN interface is switched off (GSV-6: from FWver 3.25)<br>**=6: GSV-8** CANopen NodeID (range: 1..0x7F) |
| in | *setting* | value, according to Index |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 168

Device access: Yes, CmdNo: 0x8D

date: 24.11.2015

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86setDfilterOnOff (int *ComNo*, int *Chan*, int *Type*, int *OnOff*)

Writes the enabled/disabled state of the digital FIR/IIR filter

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | If =0: Set Filter On/Off for all channels. 1..8: Set Filter On/Off for specified channel |
| in | *Type* | =0=FILT_TYPE_IIR: Set on/off state for the IIR filter.<br>=0x80=FILT_TYPE_FIR: Set on/off state for the FIR filter. |
| in | *OnOff* | If Chan=0: On/Off state in Bits<7:0>, Bit=1: Filter enabled, =0: disabled;<br>while bit 0 corresponds to channel 1, ..., bit 7 to channel 8<br>If Chan = 1..8: Set Filter on/off state for specified channel:<br>=1=GSV_TRUE: Filter enabled, =0=GSV_OK: Filter disabled |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 90

Device access: Yes, CmdNo: 0x52

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86setDfilterParams (int *ComNo*, int *Chan*, int *Type*, double * *CutRatio*, double * *Coeff*, double * *CoeffB*)

Writes all parameters for the digital FIR/IIR filter

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 0..8: If =0: Set Filter Params for all channels. Else: Set Filter Params for specified channel |
| in | *Type* | Type of filter.<br>**Bit7:** =0: IIR, =1: FIR (GSV-8 only)<br>**Bits<6:4>:** =0: Low pass, =1: high pass, =2: band pass, =3: band stop, 4: Comb (FIR only). Values 1..3: IIR only.<br>**Bits<3:0>:** Filter order (as by now, =4 for IIR, 4..14 for FIR allowed)<br>See constant macros (FILT_TYPE_<I/F>IR_<LP/HP/BP/BS>) |
| in | *CutRatio** | Pointer to double array of size=2. At array index 0, the (lower) -3dB Cutoff frequency ratio FcutOff,low/Fdata is expected. With filter types Band pass and Band stop, at array index 1, the higher -3dB Cutoff frequency ratio FcutOff,up/Fdata is expected. |
| in | *Coeff** | Pointer to double array of min. size =5. With IIR filter, the a coefficients a0..a4 are expected, with a0 at index 0.<br>With FIR filter, the first 4 FIR coefficients are expected, with a0=a7 at index 0, a1=a6 at ix.2 .. a3=a4 at ix.3. |
| in | *CoeffB** | Pointer to double array of min. size =4. Only with IIR filter, at the first 4 places the b coefficients b0..b3 are expected, with b0 at index 0. Can be NULL with FIR filter. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

The effectivity and quality of the filter is NOT tested! The user is expected to have calculated the coefficients appropriately in regard to the required cutoff frequency and filter type. Wrong coefficients can result in instable and unpredictable behavior of the filter, i.e. erratic measuring values!
In order to get safe results, use GSV86calcSetDfilterParams instead!

OrdinalNo: 96

Device access: Yes, CmdNo: 0x4E, 0x50, 0x7E

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86setDIOdirection (int *ComNo*, int *DIOgroup*, int *Direction*)

Writes the direction of the digital I/O line group

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DIOgroup* | See GSV86getDIOdirection().<br>If DIOgroup is =0, all 4 directions are set according to Bits<3:0> of Direction |
| in | *Direction* | =0 for output or =1 for input |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 153


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

Device access: Yes, CmdNo: 0x5A

date: 18.06.2015

Applicable devices: GSV-8

### int CALLTYP GSV86setDIOtype (int *ComNo*, int *DIOno*, int *DIOtype*, int *AssignedChan*)

Writes the type of the digital I/O line

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DIOno* | Number of digital IO line 0..16. 0: Set all DIO-no to the same type |
| in | *DIOtype* | See definition in GSV86getDIOtype() and annex |
| in | *AssignedChan* | Analog input channel to apply DIO functionality, if DIOtype= ThresholdSwitch (DIO_OUT_THRESHOLD_ANYVAL Bit set) or DIO_IN_TARE_SINGLE. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 154

Device access: Yes, CmdNo: 0x5C

date: 21.06.2015 10:53

Applicable devices: GSV-8

### int CALLTYP GSV86setDoutInitLevel (int *ComNo*, int *DIOno*, int *DOInitLevel*)

Writes the default level of the digital I/O line. This level is set on device boot up, eventually after a type change or if none of set output conditions are met (yet). Will be stored for all 16 I/O lines, but only meaningful for output types.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DIOno* | Number of digital IO line 0..16. 0: Set all 16 DIO levels, whereby Bit0 of DIOlevel = DIOno 1.. Bit15= DIOno 16 |
| in | *DOInitLevel* | Default out level to set: 1= high, 0=low or all 16 lines in Bits<15:0> |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). /note: Default out level is stored for corresponding DIOno(s), regardless of its type or invert state.

**Note:**

This level is set to the corresponding DIO output if:

1. The corresponding DIOtype is an output type
2. DIO initialization is called. That happens at boot-up, at type or direction changing and at parameter loading

OrdinalNo: 161

Device access: Yes, CmdNo: 0x62

date: 21.06.2015

Applicable devices: GSV-8

### int CALLTYP GSV86setDoutLevel (int *ComNo*, int *DIOno*, int *DIOlevel*)

Writes the level of digital output line

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DIOno* | Number of digital IO line 0..16. 0: Set all 16 DIO levels, whereby Bit0 of DIOlevel = DIOno 1.. Bit15= DIOno 16 |
| in | *DIOlevel* | IO level to set: 1= high, 0=low or all 16 lines in Bits<15:0> |

**Returns:**

: Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). **Note:** DIO level is set only if corresponding DIOtype= DIO_OUT_GENERALPURPOSE. Otherwise, if DIOno>0, the function returns GSV_ERROR (LastError=ERR_PAR_NOFIT_SETTINGS)

OrdinalNo: 156

Device access: Yes, CmdNo: 0x5E

date: 21.06.2015

Applicable devices: GSV-8, eventually GSV-6

### int CALLTYP GSV86setDoutThreshold (int *ComNo*, int *DIOno*, double *ThresUp*, double *ThresDown*)

Writes the two threshold values for digital output threshold switch. Will be stored for any digital I/O line, but only meaningful for digital output threshold types.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DIOno* | Number of digital IO line 1..16. |
| in | *ThresUp* | Upper threshold of threshold switch |
| in | *ThresDown* | Lower threshold of threshold switch |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

: ThresUp must be greater or equal than ThresDown

OrdinalNo: 158

Device access: Yes, CmdNo: 0x60

date: 21.06.2015

Applicable devices: GSV-8

### int CALLTYP GSV86setFrequency (int *ComNo*, double *frequency*)

Writes the measuring data frequency, with that whole measuring data frames are transmitted, it permanent value transmission is enabled (default).

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *frequency* | Number of measuring value frames per second, that the device shall transmit equidistant and permanently |


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 50

Device access: Yes, CmdNo: 0x8B

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86setFTarrayToRead (int *ComNo*, int *ArrNo*)

Sets the Force torque sensor (multi-axis) sensor calibration array for subsequent single parameter reads.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ArrNo* | Index of array to select for reading with GSV86readFTsensorCalValue. Range: 0 to N |

Remark: The maximum Array Number is the highest index where meaning ful data is stored. Call GSV86getFTsensorCalArrayInfo to determine N.

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 164

Device access: Yes, CmdNo: 0x7D

date: 05.10.2015

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86setFTsensorActive (int *ComNo*, int *OnOff*)

Sets the enabled/disabled state of the Force torque sensor (multi-axis) sensor. If *OnOff* =1, this function initializes the multi-axis sensor measurement.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *OnOff* | =1 to enable Six-axis sensor, =0 to disable it |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 135

Device access: Yes (Cmd. 0x26, 0x27)

date: 30.04.2015

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86setFTsensorActiveCalArray (int *ComNo*, int *ArrNo*)

If Force torque sensor (multi-axis) sensor calibration arrays are stored, the active one that is/will be used can be set with this function.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ArrNo* | Index of six-axis calibration array (beginning with 0) to activate. Range: 0 to 4. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:** Does not initialize the new sensor calibration data! Do this by calling GSV86setFTsensorActive

OrdinalNo: 163

Device access: Yes, CmdNo: 0x55

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86setInterfaceOnOff (int *ComNo*, int *IntNo*, int *OnOff*)

The device may support several communication interfaces. This sets enabled/disabled state of particular fieldbus/interface.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *IntfNo* | Index in linked list of interface descriptors/settings: Range: 0..(Number of interfaces - 1). E.g. use GSV86readBasicInterfSettings to find it out. |
| in | *OnOff* | =1 switch on, =0: switch off |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 177

Device access: Yes, CmdNo: 0x01, 0x7B, 0x7C

date: 24.01.2016

Applicable devices: GSV-8

### int CALLTYP GSV86setInType (int *ComNo*, int *Chan*, int *InType*)

Writes analog input type

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Number of input channel, 1..8 |
| in | *InType* | Input type, defined as follows:<br>INTYP_BRIDGE_US875 0 Differential Bridge Input at Vexcitation=8,75V<br>INTYP_BRIDGE_US5 1 Differential Bridge at Vexcitation=5V<br>INTYP_BRIDGE_US25 2 Differential Bridge at Vexcitation=2,5V<br>INTYP_SE10 3 Single-ended input+-10V<br>INTYP_PT1000 4 Temperature-Sensor PT1000 input<br>INTYP_TEMP_K 5 Temperature-Sensor Type K, absolute<br>INTYP_TEMP_K_DT 6 Temperature-Sensor Type K, relative |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 68

Device access: Yes, CmdNo: 0xA3

date: 11-18-2014

Applicable devices: GSV-8

### int CALLTYP GSV86setMeasValProperty (int *ComNo*, int *PropType*)

Writes special measuring value state, which is temporarily (not stored in non-volatile memory).

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *PropType* | **=0:** Set special value properties off, so that normal measuring mode is enabled again<br>**=1:** Set simulated value on, so that all readouts of all channels become equal to half of the full-scale value ("calibration jump")<br>**=2:** Load Zero calibration values, so that measuring values give the no-load sensor deviation.<br>**=3:** Load Zero calibration values of first six-axis sensor, so that measuring values give the (no-load) sensor deviation. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:** This state is volatile (not saved in EEPROM memory)

OrdinalNo: 124

Device access: Yes (Cmd. 0x35)

date: 3-9-2015

Applicable devices: GSV-8

### int CALLTYP GSV86setMode (int *ComNo*, unsigned long *Mode*)

Writes devices mode flags (see GSV86getMode)

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Mode* | Flags for device operation (channel independent or valid for all channels) |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 34

Device access: Yes, CmdNo: 0x27

date: 10-31-2014

Applicable devices: GSV-8, GSV-6, but different Mode-Flags

### int CALLTYP GSV86setModeAfilterAuto (int *ComNo*, int *OnOff*)

Writes the enabled/disabled setting of the automatic analog filter determination. If

enabled, the device sets the analog input filter, depending on the device data frequency.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *OnOff* | =1 to enable Automatic selection of Analog (anti-aliasing-) filter according to the configured data rate, =0 to disable it |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 137

Device access: Yes (Cmd. 0x26, 0x27)

date: 30.04.2015

Applicable devices: GSV-8

### int CALLTYP GSV86setModeMaxMin (int *ComNo*, int *OnOff*)

Writes the enabled/disabled setting of the maximum and minimum values determination. If enabled, every measuring value is compared to the maximum and the minimum value stored, and updated accordingly.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *OnOff* | =1 to enable Maximum / Minimum measuring value determination. Result can be retrieved with GSV86getMaxMinValue. =0: to disable Maximum / Minimum mode |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 146

Device access: Yes (Cmd. 0x26, 0x27)

date: 19.05.2015 22:04

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86setModeNoiseCut (int *ComNo*, int *OnOff*)

Writes the enabled/disabled setting of the noise-cut filter. If enabled, the device sets values between the noise-cut threshold and its negation (i.e. around zero) to exactly zero.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *OnOff* | =1 to enable Six-axis sensor, so that measuring values between NoiseCutThreshold and -NoiseCutThreshold are set to 0. =0: to disable Noise-cut threshold |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 139

Device access: Yes (Cmd. 0x26, 0x27)

date: 30.04.2015

Applicable devices: GSV-8

### int CALLTYP GSV86setNoiseCutThreshold (int *ComNo*, int *Chan*, double *Thres*)

If corresponding flag in Mode is set (See also: GSV86setMode), the device will set measuring values below this threshold (but above -1*threshold) to the zero value.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 0..8: Set noise cut threshold for specified channel. 0 means: all channels to the same value |
| in | *Thres* | Noise cut threshold in percentage of the full scale range. Value range is: 0.001% to 20% |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 120

Device access: Yes, CmdNo: 0x95

date: 10-31-2014

Applicable devices: GSV-8

### int CALLTYP GSV86setPassword (int *ComNo*, char * *password*)

Some write functions require the user-password / ID to be set in the device. Do this by calling this function.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *NewPW* | Char-String to user password. Default is: "Beln" for GSV-8 |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 52

Device access: Yes, CmdNo: 0x19

date: 18.06.2015

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86setTXmode (int *ComNo*, int *Index*, unsigned long *Txmode*)

Writes settings on measuring value transmission

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Index* | **=0:** Set TXmode-flags. **=1:** Set measuring value type. |
| in | *TXmode* | **If Index=0** (set TXmode-flags), defined as follows:<br>Bit0: =1: Measuring value transmission is temporarily stopped (not stored in non-volatile memory)<br>Remark: Set this easier by calling GSV86stopTX.<br>=0: Permanent measuring value transmission active (stored in non-volatile memory). |

| | | |
|---|---|---|
| | | Remark: Set this easier by calling GSV86startTX.<br>Bit1: =1: Measuring value transmission is stopped (stored in non-volatile memory)<br>=0: Permanent value transmission is active (stored in non-volatile memory)<br>Remark: Write access bits are read-only.<br>**If Index=1** (set measuring value type), defined as follows:<br>=1: Measuring value is a 16-Bit integer in binary offset format, unscaled raw value<br>Remark: Set this easier by calling GSV86setValDataType<br>=2: Measuring value is a 24-Bit integer in binary offset format, unscaled raw value<br>=3: Measuring value is a IEEE754 float, 32 bit size, scaled in physical units |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 24

Device access: Yes, CmdNo: 0x81

date: 11-19-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86setUnitNo (int *ComNo*, int *Chan*, int *UnitNo*)

Write number of enumerated physical unit (see annex)

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | If =0: Set unit-no. of all channels to the same value. 1..10: Set Unit-no specified channel |
| in | *UnitNo* | Number of unit to set.<br>For assignment of codes 0..UNIT_NO_MAX to unit itself, see manual/annex.<br>special codes: 0x000000FF: User definable unit text 1 active<br>0x000000FE: User definable unit text 2 active |

Note: Does not scale the measuring values. Do this by GSV86writeUserScale.

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 82

Device access: Yes, CmdNo: 0x10

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86setUnitText (int *ComNo*, int *Chan*, int *Code*, char * *UnitText*)

Write user-defined text for physical unit

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

| | | |
|---|---|---|
| in | *Chan* | If =0: Set unit of all channels to the same value. 1..8: Set Unit of specified channel |
| in | *code* | Control value, consisting of unit type (Bits <7:0> and code page value (Bits <23:8>)<br>Bits <23:8>: Code page for interpreting UnitText. Only used if Bits<3:0>=0. Supported values:<br>ANSI_CODEPAGE =0x0000: ANSI 8-bit<br>ASCII_ONLY 0x0001: Strictly 7-Bit ASCII only (micro sign ->'u', degree sign left away or ^, per mille->0/00)<br>DOS_CODEPAGE_437 0x01B5 (437d): Windows/DOS codepage 437<br>WIN_CODEPAGE_1252 0x04E4 (1252d): Windows codepage 1252<br>Bits<7:0>: Storing method:<br>ACTIVE_UNIT_ANY =0x00: If unit text corresponds to an available fixed unit, that one is activated (UnitNo set accordingly).<br>Otherwise, user-defined unit string is written to the device at Unit-Text 1 and activated (UnitNo set to 0xFF)<br>USER_UNIT_1 =0x01: User-defined unit string is written to the device at Unit-Text 1 and activated (UnitNo set to 0xFF)<br>USER_UNIT_2 =0x02: User-defined unit string is written to the device at Unit-Text 2 and activated (UnitNo set to 0xFE)<br>SET_FIXED_UNIT =0x10: If unit text corresponds to an available fixed unit, that one is activated (UnitNo set accordingly).<br>Otherwise, nothing is set and an error is triggered.<br>WRITE_USER_UNIT_1 =0x11: User-defined unit string is written to the device at Unit-Text 1, but not activated (UnitNo not set)<br>WRITE_USER_UNIT_2 =0x12: User-defined unit string is written to the device at Unit-Text 2 but not activated (UnitNo not set) |
| in | *UnitText* | Pointer to string with Unit-Text to set. Maximum text-length is 7 chars |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Remarks:**

There're only two user-defined unit strings available on the device (length is up to 7 chars each), while the UnitNo is stored for each of the 8 input channels individually.

OrdinalNo: 86

Device access: Yes, CmdNo: 0x0F, 0x10 or 0x12

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

## int CALLTYP GSV86setValDataType (int *ComNo*, int *Datatype*)

Write the measuring values data type

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Datatype* | Data type enumeration, defines as follows:<br>DATATYP_INT16 1 Datatype sent by GSV-8 is 16-Bit raw value in binary offset format<br>DATATYP_INT24 2 Datatype sent by GSV-8 is 24-Bit raw value in binary offset format (GSV-8 only)<br>DATATYP_FLOAT 3 Datatype sent by GSV-8 is 32-Bit Float, |

| | | |
|---|---|---|
| | | scaled in physical units |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 62

Device access: Yes, CmdNo: 0x81

date: 11-19-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86setZero (int *ComNo*, int *Chan*)

Perform a tare routine, so that measuring values become zero

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | if =0: Set all channels to zero. 1..8: Set specified channel to zero. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 56

Device access: Yes, CmdNo: 0x0C

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86simulateDfilter (int *ComNo*, int *Analyze_Ftyp*, double *StartVal*, double *EndVal*, double *Fa*, int *Points*, char * *Filepath*)

Simulate digital FIR/IIR filter

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Analyze_Ftyp* | one of<br>SIMUL_DFILT_FREQ_RESPONSE get Frequency response<br>SIMUL_DFILT_STEP_RESPONSE get step response<br>To simulate FIR filter, OR with FILT_TYPE_FIR |
| in | *StartVal* | First frequency if Analyze_Ftyp=SIMUL_DFILT_FREQ_RESPONSE<br>Start value (before step) if Analyze_Ftyp=SIMUL_DFILT_STEP_RESPONSE (standard step: =0) |
| in | *EndVal* | Last frequency if Analyze_Ftyp=SIMUL_DFILT_FREQ_RESPONSE<br>Step value (after step) if Analyze_Ftyp=SIMUL_DFILT_STEP_RESPONSE (standard step: =1) |
| in | *Fa* | Sampling frequency = Data rate |
| in | *Points* | Number of simulated points (frequencies or samples, respective) |
| in | *Filepath* | Full path to file, where results are written to. File should not exist before; otherwise, it will be overwritten. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Additional possible error codes:
DF_ERR_NOT_INIT 0x30000201 GSV86calcSetDfilterParams not called before
DF_ERR_OPT_WRONG 0x30000202 wrong parameters
DF_ERR_NO_CONVERGENCE 0x30000203 coefficient calculation failed to converge
DF_ERR_COEFF_SUM_TOOBIG 0x30000208 coefficients are too big

**Note:**

GSV86calcSetDfilterParams must be called before. There, Chan can be set to -1 for simulation only (no device access).

OrdinalNo: 99

Device access: No

date: 3-6-2015

### int CALLTYP GSV86startTX (int *ComNo*)

Starts permanent measuring value transmission, if it was stopped before.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: starts GSV-4 value transmission. State not saved in device non-volatile memory.

OrdinalNo: 16

Device access: Yes, CmdNo: 0x24

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86stopTX (int *ComNo*)

Stops permanent measuring value transmission

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: stops GSV-8 value transmission. State not saved in device's non-volatile memory.

OrdinalNo: 18

Device access: Yes, CmdNo: 0x23

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86storeSettings (int *ComNo*, int *DataSetNo*)

Saves all device configuration parameters, except those mentioned in description for GSV86loadSettings()

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DataSetNo* | 2..6: Store settings at specified data set (GSV-8 only) |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 78

Device access: Yes, CmdNo: 0x0A

date: 10-31-2014

Applicable devices: GSV-8

### int CALLTYP GSV86switchBlocking (int *ComNo*, int *OnOff*)

Enabled or disables parameter writes on device

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *OnOff* | =0: Disable write protection, =1: Enable write protection |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

Precondition: Device password (=User-ID) set (see GSV86setPassword() )
OrdinalNo: 150

Device access: Yes, CmdNo: 0x92

date: 16.04.2015

Applicable devices: GSV-8

### int CALLTYP GSV86triggerValue (int *ComNo*)

Trigger transmission of one measuring data frame, if permanent value transmission is off.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

Retrieve measuring value frame with GSVread

OrdinalNo: 144

Device access: Yes, CmdNo: 0x3B

date: 16.04.2015

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86writeAnalogFilterCutOff (int *ComNo*, double *CutOffFreq*)

Writes -3dB cut-off frequency of analog input filter.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *CutOffFreq* | Desired cutoff frequency of analog input filter. |

| | | |
|---|---|---|
| | | Set to 0, to enable automatic cutoff frequency setting (according to data rate).<br>Otherwise, only 3 fixed cutoff frequency values will be set, closest to parameter CutOffFreq:<br>If CutOffFreq <= FDATA_MAX_FG28HZ: ANAFILT_LOW will be set, otherwise:<br>If CutOffFreq <= FDATA_MAX_FG1KHZ: ANAFILT_MID will be set. See macros above. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed.
GSV_TRUE, if cutoff frequency is set by device automatically, according to the data rate (see below).
GSV_OK, if automatic filter setting is disabled or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 128

Device access: Yes, CmdNo: 0x26, 0x27 or 0x91

date: 16.04.2015

Applicable devices: GSV-8

### int CALLTYP GSV86writeAnalogOutOffset (int *ComNo*, int *Chan*, double *Offset*)

Write Offset value of analog output. This voltage (or current) then represents a measuring value =0.
The analog output scaling value is adjusted as follows: Scaling= FSval*(1-(Offset/100))

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: Channel number of analog output to change offset |
| in | *Offset* | Value in percentage of positive full-scale value FSval. Range: -60%..60% FSval is: Output type: 0 & 1: 10V 2 & 3: 5V 4: 12mA 6: 10 mA |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 104

Device access: Yes, CmdNo: 0x05

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86writeAnalogOutScale (int *ComNo*, int *Chan*, double *Scale*)

Writes the user-defined scaling value, with which the analog output can be scaled to get a specific output value a specific input (GSV-8). With GSV-6, this value scales the input range down to a desired input sensitivity.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: Channel number of scale value to write |
| in | *Scale* | Desired scaling value. This is a factor representing the deviation of the output range based on the type. See GSV86readAnalogOutScale(). |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed

error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 108

Device access: Yes, CmdNo: 0x07

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86writeAoutDirect (int *ComNo*, int *Chan*, int *Code*)

Writes value for the analog output, if it is in direct-mode, i.e. decoupled from analog input

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: Channel number of analog output to set |
| in | *Code* | 16-Bit DAC code, nominally as follows:<br>0x0000: -11 V for -10..10 and 0..10 V output type<br>0 mA for current output<br>0x8000: 0V for voltage output<br>0xFFFF: 11 V or 5.5V for voltage, 24mA for current output |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

Corresponding analog output must be in direct mode. See GSV86setAnalogOutType *mode* parameter
The formula for calculation of *Code* depends on the Out type (see GSV86setAnalogOutType *type* parameter):
Type 0 and 1 (0..10V, -10..10V): $Code = (Uout \cdot 2978.91) + 32768$

Type 2 and 3 (0..5V, -5..5V): $Code = (Uout \cdot 5957.82) + 32768$

Type 4 and 6 (4..20mA, 0..20mA): $Code = Iout[mA] \cdot 2730.667$

Whereby *Uout* is the output voltage in V and *Iout* the output current in mA.
*Code* should be rounded to the nearest integer value, its value range is from 0 to 65535.

OrdinalNo: 110

Device access: Yes, CmdNo: 0x08

date: 10-31-2014

Applicable devices: GSV-8

### int CALLTYP GSV86writeDeviceHours (int *ComNo*, double *Hours*)

Writes the user-definable device hours counter
Precondition: User-ID set.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Hours* | Hours to set. Device will increment value by minutes and store in non-volatile memory. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

OrdinalNo: 117

Device access: Yes, CmdNo: 0x57

date: 26-02-2015

Applicable devices: GSV-8

### int CALLTYP GSV86writeFTsensorCalArray (int *ComNo*, int *ArrNo*, const char * *SensorSerNo*, double *MatrixNorm*, double *InSens*, double * *Matrix*, double * *Offsets*, double * *MaxVals*, double * *Zvals*)

Writes Force-torque (multi-axis) sensor parameters. These are complete for sensors using Matrix A (1st order) only. Otherwise, call GSV86writeFTsensorCalArrExt first.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ArrNo* | Bits <6:0>: Calibrations struct index. Used for storing several sensors/calibration data sets, consecutive and beginning with 0 |
| in | *SensorSerNo* | pointer to String of 8 number digits: Sensor serial number read from here |
| in | *MatrixNorm* | single double value: Scaling of calibration matrix. =1 if matrix values are scaled in N/mV/V and Nm/mV/V respectively. |
| in | *InSens* | single double value: input sensitivity of origin of calibration matrix. =1 if matrix values are scaled in N/mV/V and Nm/mV/V respectively. |
| in | *Matrix* | ptr to array of 36 double values of the Calibration matrix, Index order: first row 0..5, 2nd row 6..11... |
| in | *Offsets* | ptr to array of 3 double values: mechanical offsets of sensor installation in meters 0..2: Ox,Oy,Oz |
| in | *MaxVals* | ptr to array of 6 double values: maximum values for the 6 outputs. Order: 0..5: Fx,Fy,Fz,Mx,My,Mz |
| in | *Zvals* | ptr to array of 6 double values: No-load sensor deviations of the 6 inputs. Order: 0..5: Fx,Fy,Fz,Mx,My,Mz (GSV-8 only) |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Precondition**: Device Password (=User-ID) set (see GSV86setPassword() )

Note: Does not initialize the new sensor calibration data! Do this by calling GSV86setFTsensorActive

OrdinalNo: 44

Device access: Yes, CmdNo: 0x48, 0x7F

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86writeFTsensorFromFile (int *ComNo*, int *ArrNo*, const char * *DatFilePath*)

Writes all Force-torque (multi-axis) sensors parameters, read from the calibration files (*.dat and *.matrix)

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ArrNo* | 0x80: TEDS storage (coming hardware versions), Bits <6:0>: Calibrations structs index for storing several sensors/calibration data |

| | | |
|---|---|---|
| | | sets |
| in | *DatFilePath* | String (ptr) to full path of sensor calibration file: "<drive>:\\<path\\<name>.dat" |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

Note: Does not initialize the new sensor calibration data! Do this by calling GSV86setFTsensorActive

**Precondition**: Device Password (=User-ID) set (see GSV86setPassword() )

OrdinalNo: 100

Device access: Yes, CmdNo: 0x48, 0x7F

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86writeFTsensorGeoOffsets (int *ComNo*, int *ArrNo*, double * *offsets*)

Writes mechanical distance offsets for force/torque sensor, i.e. the distances between the point for measuring the torque values and the sensor origin.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ArrNo* | Bits <6:0>: Calibrations structs index for storing several sensors/calibration data sets |
| in | *offsets* | pointer to double array, which contains the three offsets values, whereby at index 0: offset in meters in X direction, index 1: offset in meters in Y direction, index 2: offset in meters in Z direction. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Precondition**: Device password (=User-ID) set (see GSV86setPassword() )

Note: Does not initialize the new sensor calibration data! Do this by calling GSV86setFTsensorActive

OrdinalNo: 45

Device access: Yes, CmdNo: 0x48, 0x7F

date: 25.04.2015

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86writeInterfaceBaud (int *ComNo*, int *IntNo*, int *Baud*)

Writes the communication bit rate for specified interface. Valid after rebooting or (re-) activating the interface. Can not be used with USB interface.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *IntfNo* | Index in linked list of interface descriptors/settings: Range: 0..(Number of interfaces - 1). May use GSV86readBasicInterfSettings to find it out. |
| in | *Baud* | baud rate in Bits/s |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 178

Device access: Yes, CmdNo: 0x01, 0x7B, 0x7C

date: 24.01.2016

Applicable devices: GSV-8, eventually GSV-6

### int CALLTYP GSV86writeInterfaceSetting (int *ComNo*, int *Ix*, unsigned long *Data*)

Writes settings for communication interface

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Ix* | Index in linked list of interface descriptors/settings:<br>if Ix = 0..(Number of interfaces - 1) =Interface Number: Basic settings. Otherwise: extended settings |
| in | *Data* | if Ix = 0..(Number of interfaces - 1): Flag value<br>Else: Value, according to Type of Data (e.g. read with GSV86readInterfaceSetting), see Annex, page 116 |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 176

Device access: Yes, CmdNo: 0x7C

date: 24.01.2016

Applicable devices: GSV-8, eventually GSV-6

### int CALLTYP GSV86writeTEDSrawData (int *ComNo*, int *Chan*, const unsigned char * *DataIn*, int *NumBits*, int *StartBitAdr*)

Writes binary raw data into TEDS (transducer electronic data sheet) sensor

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Input channel No (1..8) with TEDS sensor connected |
| in | **DataIn* | Array with data to write. Must be byte-aligned at base. Size=NumBits/8, rounded up. |
| in | *NumBits* | Number of Bits to write |
| in | *StartBitAdr* | Start Bit address of 1-wire EEPROM. TEDS-checksum not included. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

1. DataIn must not contain the Checksum Byte(s); the device itself will generate them.

2. Device password (=User-ID) must be set (see GSV86setPassword() )

OrdinalNo: 184

Device access: Yes, CmdNo: 0x66, 0x67

date: 06.07.2016

Applicable devices: GSV-8

## int CALLTYP GSV86write1wire (int *ComNo*, int *Chan*, const unsigned char * *DataIn*, int *NumBytes*, int *StartByteAdr*)

Writes binary raw data into 1-wire memory device

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Input channel No (1..8) with 1-wire memory connected |
| in | **DataIn* | Array with data to write. Must be byte-aligned at base. Size=Numbytes |
| in | *NumBytes* | Number of Bytes to write |
| in | *StartByteAdr* | Start Byte address of 1-wire EEPROM. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

1. From GSV-8 main FW-ver. 1.53 and Co-MCU version 2. Fails otherwise
2. Device password (=User-ID) must be set (see GSV86setPassword() )

OrdinalNo: 245

Device access: Yes, CmdNo: 0x66, 0x67

date: 17/01/2020

Applicable devices: GSV-8

## int CALLTYP GSV86read1wireTemperatur (int *ComNo*, int *Chan*, *double* temp*)

Reads temperature value from 1-wire temperature sensing device

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Input channel No (1..8) with 1-wire sensor connected |
| out | **temp* | Temperature in °C written, if successful |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note:**

1. From GSV-8 main FW-ver. 1.53 and Co-MCU version 2. Fails otherwise.

OrdinalNo: 246

Device access: Yes, CmdNo: 0x65

date: 17/01/2020

Applicable devices: GSV-8

### int CALLTYP GSV86writeUserOffset (int *ComNo*, int *Chan*, double *Offset*)

If the measuring value data type is set to Float (see GSV86getValObjectInfo), the device can add a user-defined offset to every measuring value. This function writes it.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | if =0: Set offset of all channels to the same value. 1..10: Read offset of specified channel |
| in | *Offset* | Offset value, which is added by the device to every measuring value, if the device's measuring value data type is DATATYP_FLOAT |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 74

Device access: Yes, CmdNo: 0x9B

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86writeUserScale (int *ComNo*, int *Chan*, double *Norm*)

The device can be parametrized to have measuring values scaled in physical units. This function writes that scale factor. It is normally not used with Six-axis and temperature sensor measuring.

If the measuring value data type is set to one of the integer types (see GSV86getValObjectInfo), values retrieved by GSV86read or GSV86readMultiple should be multiplied with the User Scale by the caller in order to get physically scaled values.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Number of input channel to set scaling factor. Range: 0..10. 0: set all channels. |
| in | *Norm* | Scaling factor. |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed
If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 30

Device access: Yes, CmdNo: 0x15

date: 10-31-2014

Applicable devices: GSV-8, GSV-6

### int CALLTYP GSV86writeZeroValue (int *ComNo*, int *Chan*, int *zero*)

Direcly change the raw value offset.

**Parameters:**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | 1..8: channel to write zero value |
| in | *zero* | Zero raw value, represented as a signed integer value of 16 or 24 bits magnitude, depending on the measuring value type |

**Returns:**

Simple error code: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Note**: Device password (=User-ID) must be set (GSV-8).

OrdinalNo: 142

Device access: Yes, CmdNo: 0x03

date: 16.04.2015

Applicable devices: GSV-8, GSV-6

---

## int CALLTYP GSV86BTexitConfig (int *ComNo*)

Exit GSV-6/ITA Bluetooth configuration API

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: OrdinalNo: 401

Device access: Yes, BGscript, CmdNo: 0xE1

date: 26.05.2017

Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTgetBTmode (int *ComNo*, int * *BTmode*)

Read GSV-6/ITA Bluetooth mode

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *BTmode* | =0: Read Radio power for Bluetooth LE (low energy) =1: Read Radio power for Bluetooth Classic |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: OrdinalNo: 404

Device access: Yes, BGscript, CmdNo: 0xE6

date: 26.05.2017

Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTgetGSVonOff (int *ComNo*, int * *OnOff*)

Read GSV-6/ITA on or off state via Bluetooth

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *OnOff* | =0: Switch measuring amplifier off =1: Switch measuring amplifier on |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed

error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: OrdinalNo: 409

Device access: Yes, BGscript, CmdNo: 0xED

date: 26.05.2017

Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTgetInType (int *ComNo*, int *chan*, int * *val*)

Read GSV-6BT/ITA input Type

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *chan* | Input channel (2..6) to read type |
| out | *val* | =0: wheatstone bridge input =1: single ended voltage input |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 413

Device access: Yes, BGscript, CmdNo: 0xF0

date: 26.05.2017

Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTgetLoggerInterval (int *ComNo*, int * *interval*)

Get Alarm Interval for measuring value logging to SD-card.

**Remarks**

Reads value from BGscript only; for full Logger Intervall inquiry, please use GSV86getLoggerAlarmInterval

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *interval* | Interval in seconds, if return value =1 |

**Returns**

: =0: Interval in seconds, written to interval =1: Interval in minutes, here unknown; or Simple errorcode: GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 416

Device access: Yes, BGscript, CmdNo: 0xF3

date: 11.01.2018

Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTgetMaxPower (int *ComNo*, int *BTmode*, int * *pwr*)

Read GSV-6/ITA Bluetooth maximum radio transmission power

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *BTmode* | =0: Read Radio power for Bluetooth LE (low energy) =1: Read Radio power for Bluetooth Classic |
| out | *pwr* | If BTmode=0 (LE): 0: 7dBm (default) 1: 2dBm 2: -3dBm 3: -8dBm 4: -13dBm 5: -18dBm If BTmode =1 (classic): Value Range -20 ... +12. Default: 12 |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: OrdinalNo: 402

Device access: Yes, BGscript, CmdNo: 0xE2 or 0xE4

date: 26.05.2017
Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTinitConfig (int *ComNo*, int * *BGversion*)

Initialize GSV-6/ITA Bluetooth configuration API

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *BGversion* | If not NULL, version of BG-script. Bits<15:8>: Main-version number. Bits<7:0>: Revision number |

**Returns**

Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: OrdinalNo: 400

Device access: Yes, BGscript, CmdNo: 0xE0

date: 26.05.2017
Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTreadBatteryVoltage (int *ComNo*, double * *Voltage*)

Read Module battery voltage

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *Voltage* | Battery voltage in volts |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: OrdinalNo: 412

Device access: Yes, BGscript, CmdNo: 0xEA

date: 29.05.2017 \revision 1.25: return value corrected.
Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTreadName (int *ComNo*, char * *Name*)

Read GSV-6/ITA Bluetooth device name

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *Name* | NULL-terminated String: Bluetooth device name. |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 406

Device access: Yes, BGscript, CmdNo: 0xE9

date: 26.05.2017
Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTreset (int *ComNo*, int *ResetType*)

Reset GSV-6/ITA Bluetooth module

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ResetType* | Normal reset oder boot to DFU mode.Parameter: 0: Normal,1: DFU |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: OrdinalNo: 408

Device access: Yes, BGscript, CmdNo: 0xEC

date: 26.05.2017
Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTsetBTmode (int *ComNo*, int *BTmode*)

Set GSV-6/ITA Bluetooth mode

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *BTmode* | =0: Read Radio power for Bluetooth LE (low energy) =1: Read Radio power for Bluetooth Classic |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: OrdinalNo: 405

Device access: Yes, BGscript, CmdNo: 0xE7

**Remarks**

: Warning: If device is set to Bluetooth LE, serial communication may not possible anymore, since BT Le doesn't support the SPP profile.

Date: 26.05.2017
Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTsetDefault (int *ComNo*)

Load and set GSV-6/ITA Bluetooth communication default settings

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 411
Device access: Yes, BGscript, CmdNo: 0xEF
date: 26.05.2017

Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTsetDigitalOut (int *ComNo*, int *DIOno*, int *OnOff*)

Set GSV-6BT/ITA digital output

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DIOno* | Number of digital output. Range: 1..4. Nr. 4 is the build-in free LED |
| in | *OnOff* | =0: Digital out Low / LED off =1: Digital out High / LED on |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 415
Device access: Yes, BGscript, CmdNo: 0xF1
date: 26.05.2017
Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTsetGSVonOff (int *ComNo*, int *OnOff*)

Switch GSV-6/ITA on or off via Bluetooth

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *OnOff* | =0: Switch measuring amplifier off =1: Switch measuring amplifier on |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: OrdinalNo: 410

Device access: Yes, BGscript, CmdNo: 0xEE

date: 26.05.2017
Applicable devices: GSV-6BT

---

## int CALLTYP GSV86BTsetInType (int *ComNo*, int *chan*, int *val*)

Set GSV-6BT/ITA input Type

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *chan* | Input channel (2..6) to read type |
| in | *val* | =0: wheatstone bridge input =1: single ended voltage input |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 414
Device access: Yes, BGscript, CmdNo: 0xEB
date: 26.05.2017
Applicable devices: GSV-6BT


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

## int CALLTYP GSV86BTsetLoggerInterval (int *ComNo*, int *interval*)

Set Alarm Interval for measuring value logging to SD-card.

**Remarks**

: Writes seconds value to BGscript only; for full Logger Intervall access, please use GSV86setLoggerAlarmInterval

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *intervSec* | =0: Interval in Minutes (triggered by RTC) or 3..59: Interval in seconds |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 417
Device access: Yes, BGscript, CmdNo: 0xF2
date: 11.01.2018
Applicable devices: GSV-6BT

## int CALLTYP GSV86BTsetMaxPower (int *ComNo*, int *BTmode*, int *pwr*)

Set GSV-6/ITA Bluetooth maximum radio transmission power

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *BTmode* | =0: Set Radio power for Bluetooth LE (low energy) =1: Set Radio power for Bluetooth Classic |
| in | *pwr* | If BTmode=0 (LE): 0: 7dBm (default) 1: 2dBm 2: -3dBm 3: -8dBm 4: -13dBm 5: -18dBm If BTmode =1 (classic): Value Range -20 ... +12. Default: 12 |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: OrdinalNo: 403

Device access: Yes, BGscript, CmdNo: 0xE3 or 0xE5

date: 26.05.2017
Applicable devices: GSV-6BT

## int CALLTYP GSV86BTwriteName (int *ComNo*, char * *Name*)

Write GSV-6/ITA Bluetooth device name

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Name* | NULL-terminated String of ASCII-only chars with maximum length =15 characters. |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark:

With Bluetooth LE, the name will be written into the GAP service database. OrdinalNo: 407

Device access: Yes, BGscript, CmdNo: 0xE9

date: 26.05.2017

Applicable devices: GSV-6BT

---

## int CALLTYP GSV86controlFileLog (int *ComNo*, int *control*)

Control GSV-6BT/ITA file logger

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *control* | 0: Finish file recording and close file 1: Open file, if necessary and write one line of actual measuring values 2: Open file, if necessary and start file recording permanently **3**: Open a new file for recording. Then, a coming trigger for single value log can be executed faster. If a file was open in append mode, this creates and opens a new file. 0x40: Reset RTC-Alarm only 0x41: Reset RTC-Alarm and restart alarm interval |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 224

Device access: Yes, CmdNo: 0x6F

date: 05.10.2017

Applicable devices: GSV-6 with Logger, e.g. GSV-6BT

---

## int CALLTYP GSV86copyFromSDfile (int *ComNo*, char * *DstPath*, char * *SrcPath*)

Copy file from SDcard in GSV-6BT/ITA file logger to file

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *DstPath* | Complete path of file to be copied to (Windows notation, starting with drive letter) |
| in | *SrcPath* | Full Path on the SD-card or file name of source file to be copied from. If full path, it must start with a backslash () as directory separator. Otherwise, complete file name only without any backslash(es). Then, directory must already be open. If NULL or empty string: Copy from file lastly written by logger (if available). |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 229

Device access: Yes, CmdNo: 0x71, 0x6F or 0x74

date: 05.10.2017

Applicable devices: GSV-6 with Logger, e.g. GSV-6BT

---

## int CALLTYP GSV86getAutoZeroProperty (int *ComNo*, int *Type*, int *ix*, int * *prop*)

Read Properties of the AutoZero function (GSV-8, FWver >= 1.39)

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

| | | |
|---|---|---|
| in | *Type* | Property type to read: =0: ApplyAZ: Channel-flags that specify which channels shall be set to zero, or (with Mode=1) for which channels the AutoZero function is enabled. =1: UseThreshold: With Mode=0: Flags, that specify, which channels shall be compared with their correspondend threshold. The boolan results are then ANDed and if all conditions meet, the SetZero routine will be performed. =2: Mode: Bit 0: =0: Channels are grouped, i.e. if all thershold and time conditions meet (see UseThreshold), the channels configured with ApplyAZ will be set to Zero. Bit 0 =1: Auto-Zero will be applied on all channels configured with ApplyAZ, independently of each other (UseThreshold ignoriered). |
| in | *ix* | reserved (ignored) |
| out | *prop* | Property flags, see description for Type. Type =0 und =1: Channel flags: Bit 0: Condition/Action applied to channel 1, and so forth until Bit 7: applied to channel 8. |

**Returns**

Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().
OrdinalNo: 237 Device access: Yes, CmdNo: 0x98
date: 17.07.2018
Applicable devices: GSV-8

## int CALLTYP GSV86getCounterFreqMode (int *ComNo*, int *CntNo*, int *index*)

Read the QEI/counter/frequency measuring mode

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *CntNo* | Number of counter (0..1). Must be =0 for GSV-6/ITA. |
| in | *index* | mode to read (see return value) |

**Returns**

If **index =0**: General mode / QEI flags. **Bit 0**: =1: Counter module is present **Bit 1**: =1: Counter measuring enabled **Bit 2**: =1: Frequency/Speed measuring enabled. With GSV-6, these two functions can't be used the same time. With GSV-6, they can be used in combination, but for counter No. 0 only. **Bits <4:3>**: QEI mode: 00 FreeRun, non-QEI mode 01 QEI x1 10 QEI x2 11 QEI x4 **Bit 5**: =0: ignore Index/Home input =1: Pulse at Index/Home input copies counter offset to counter value (not recommended with Velocity / frequency measuring) **Bit 6:** =1: Counter saturates to maximum numeric values =0: Counter doesn't saturate (rolls over with sign exchange) **Bit 7**: =0: Device doesn't store counter val in non-volatile memory. At power-up, counter is initialized with 0 =1: At power-down, Device stores actual counter val in non-volatile memory, if Counter measuring enabled **Bit 8**: =1: With Frequency/Speed measuring enabled (Bit2=1), the GSV-6 device uses an additional oszillator for more precise period measuring. GSV-8: Period measuring method. **Bit 9**: =1: Automatic selection of frequency mode: Period measuring or counter (GSV-8 only)
**Bit 10**: =0: Pullup-Resistors (10kΩ) **enabled** =1: Pullup- Resistors (10kΩ) **off** (GSV-8 only)
**Bit 11**: =1: Index Input inverted (GSV-8 only) Bit 12: =1: Use input noise filter (GSV-8 only)
If **index =1**: Frequency / velocity mode: Bits <15:0>: Gate time counter: Counts data frequency periods; if reached, samples counter value and calculates frequency / velocity value. Period measuring: Number of data periods without change, after output value is set to 0.
If **index =2**: Counter offset value Or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 192

Device access: Yes, CmdNo: 0x69

date: 22.06.2017

Applicable devices: GSV-8, GSV-6 with counter (e.g. GSV-6BT)

## int CALLTYP GSV86getLoggerAlarmInterval (int *ComNo*, int * *IntervSec*)

Get interval for slow (power-saving) SD-card Logging

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *IntervSec* | Time interval in seconds for writing a measuring value row to the file |

**Returns**

: Mode: 0: Slow interval not enabled 1: Slow interval set in seconds (managed by BGscript only) 2: Slow interval set in Minutes, managed by RTC and BGscipt Or Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 416
Device access: Yes, GSV-6 CPU and BGscript, CmdNo: 0xE0, 0xED, 0xF3, 0xE1, 0x6D, 0x6B
date: 12.01.2018
Applicable devices: GSV-6 with Logger, e.g. GSV-6BT

---

## int CALLTYP GSV86getModelInfo (int *ComNo*, int * *ExtModel*, int * *PeriMCUver*, int * *reserved*)

Read Device Model information

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *ExtModel* | Extended Model Enumeration, e.g. =0: no special periphery, =1: GSV-6BT Model |
| out | *PeriMCUver* | Version of perphery processor software (if ExtModel>0), e.g. BGversion of GSV-6BT Model |
| out | *reserved* | not yet used |

**Returns**

: Basic Model: =6 for GSV-6 or =8 for GSV-8, or =0: unknown or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().
OrdinalNo: 243
Device access: No
date: 22.09.2019

---

## int CALLTYP GSV86getSDfileInfo (int *ComNo*, int * *DirLevel*, int * *Flags*, unsigned long * *Size*, char * *Changed*)

Get information about a file or directory on GSV-6BT/ITA SD-card, that was lastly opened with GSV86openSDfileDir or queried by GSV86querySDfileSys

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *DirLevel* | Actual directory level. |
| out | *Flags* | Bit 0: =1 Item is a directory. =0: It's a file Bit 1: =1: Read only flag set Bit 2: =1: Hidden flag set Bit 3: =1: System flag set Bit 6: =1: Archive flag set |
| out | *Size* | File size in bytes. =0, if it's a directory |
| out | *Changed* | Array of 6 unsigned char values, where file/dir time information |

| | | |
|---|---|---|
| | | "last changed" is written to. Changed[0]: Year since 2000 (1..107) Changed[1]: Month of year 1..12 Changed[2]: Day of Month 1..31 Changed[3]: Hour 0..23 Changed[4]: Minute 0..59 Changed[5]: Seconds 0..58 (always even) |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: OrdinalNo: 227

Device access: Yes, CmdNo: 0x72

date: 06.10.2017
Applicable devices: GSV-6 with Logger, e.g. GSV-6BT

---

## int CALLTYP GSV86openSDfileDir (int *ComNo*, char * *Name*, int *Flags*)

Open a file on GSV-6BT/ITA SD-card for reading in directory lastly openend

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Name* | Name string of item to open. Must be in 8.3 format for files or 8 chars long for directories or shorter. Always 0-terminated. |
| in | *Flags* | **Bit 0**: =1 Open a directory. =0: Open a file **Bit 1**: =1: Open last file/dir logged to = -1: Close File |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: 1. The device adds new directory or file names to the entire path. The length of the whole path name must not exceed 255 characters.

1. It's not necessary to use GSV86openSDfileDir for copying a file with GSV86copyFromSDfile.
2. After opening a directory successfully, the device increments the DirLevel, used with GSV86querySDfileSys and GSV86getSDfileInfo.

OrdinalNo: 226

Device access: Yes, CmdNo: 0x71

date: 06.10.2017
Applicable devices: GSV-6 with Logger, e.g. GSV-6BT

---

## int CALLTYP GSV86querySDfileSys (int *ComNo*, int *Ctrl*, int *DirLevel*, char * *Name*, int * *Flags*)

Search GSV-6BT/ITA SD-card file system

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Ctrl* | Control value: =0: Start, e.g. at root directory =1: Query next entry =0xFE: Read directory name, that data logger has created lastly =0xFF: Read file name, where last values were written by data logger |
| in | *DirLevel* | Directory level. Start with ItemNo=0 and DirLevel=0 to open the root directory. Query all items in the same directory (e.g. root with DirLevel=0). After opening another directory with GSV86openSDfileDir, increment DirLevel to query items within that one. Range: 0..254 |
| out | *Name* | If a file or directory is found, the Name in 8.3 format is written into |

| | | |
|---|---|---|
| | | Name. Must be at least of 13 bytes in size. Always 0-terminated. |
| out | *Flags* | **Bit 0**: =1 Item found is a directory. =0: It's a file **Bit 1**: =1: Name recognized, i.e. item found was probably created by GSV-6/ITA file logger **Bit 7**: =1: No item found (anymore). In that case, also the Name string is empty. |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 225

Device access: Yes, CmdNo: 0x70

date: 06.10.2017

Applicable devices: GSV-6 with Logger, e.g. GSV-6BT

---

## int CALLTYP GSV86readAutoZeroSetting (int *ComNo*, int *Type*, int *Chan*, double * *val*)

Read Settings of the AutoZero function (GSV-8, FWver >= 1.39)

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Type* | Parameter type to read: =0: Read Auto-Zero count period in seconds (Chan ignored) =1: Read Auto-Zero threshold for Chan |
| in | *Chan* | Channel index to read (1..8), if Type=1 |
| out | *val* | Period or threshold read |

**Returns**

Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().
OrdinalNo: 235
Device access: Yes, CmdNo: 0x96
date: 17.07.2018
Applicable devices: GSV-8

---

## int CALLTYP GSV86readDevCalInfo (int *ComNo*, unsigned char * *CalTime*, char * *OpName*)

Read Information about the last manufacturer's device calibration (GSV-8, FWver >= 1.40)

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *CalTime* | Pointer to array of 4 bytes containing the date of the last calibration: CalTime[0]: Year since 2000 (1..99 /255) CalTime[1]: Month of year 1..12 CalTime[2]: Day of Month 1..31 CalTime[3]: Hour of day 0..23 |
| out | *OpName* | Array of 8 chars: Name of the institution of the last device calibration and/or operatos's acronym. |

**Returns**

Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().
OrdinalNo: 239
Device access: Yes, CmdNo: 0x1D, 0x21
date: 17.07.2018
Applicable devices: GSV-8


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

## int CALLTYP GSV86readFTsensorCalArrExt (int *ComNo*, int * *Styp*, double * *MatrixB*, int * *Fact1ix*, int * *Fact2ix*, char * *ModelName*)

Read extended calibration values for six-axis sensor calibration (GSV-8, FWver >= 1.39)

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *Styp* | Bits<7:0>: Sensor type: =0: Six-Axis, =1: Six-Axis with MatrixB (2nd order), =2: 3/4 axis sensoor (reserved) Bits<15:8>: Reserved for special calculation type, e.g. Bit 1 (mask 0x100): Way calculation |
| out | *MatrixB* | ptr to array of 36 double values: Calibration matrixB (2nd order) will be read, order: first row 0..5, 2nd row 6..11... Used only if Styp<7:0> =1. |
| out | *Fact1ix* | ptr to array of 6 int values, that are the indices on the raw input value array, used as the 1st factor that form the vector thet the MatrixB is multiplied with. |
| out | *Fact2ix* | ptr to array of 6 int values, that are the indices on the raw input value array, used as the 2nd factor that form the vector thet the MatrixB is multiplied with. |
| out | *ModelName* | ptr to String of maximum of 16 chars that contains the model name of the sensor |

**Returns**

Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remarks:
1. MatrixB, Fact1ix and Fact2ix values are read only if Styp<7:0> =1.

1. For MatrixB, Fact1ix, Fact2ix and ModelName the ptr-values can be NULL. Values are not read in that case.

OrdinalNo: 233

Device access: Yes, CmdNo: 0x47, 0x1B
date: 17.07.2018
Applicable devices: GSV-8

## int CALLTYP GSV86readLoggerSettings (int *ComNo*, int *index*, unsigned long * *value*)

Read GSV-6BT/ITA file logging settings

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *index* | **Index 0:** Basic Flags:<br>Bit<0>: =1: Suitable SD card inserted<br>Bit<1>: =1: File actually open for recording (read-only)<br>Bit<2>: =1: Recording-enabled-state (persistently stored)<br>Bit<4>: =1: Permanent data recording<br>=0: Single value logging<br>Bit<5>: =1: Recoding by trigger condition (by now, digital input only)<br>**Index 1:** Bit<0>: =1: Alarm Mode: RTC activates alarm-line as configured by alarm interval<br>**Index 2:** Bit<0>: Directory creation mode:<br>Determines, at which time a new directory will be created, when the recording is started: |

| | | |
|---|---|---|
| | | =0: New directory every day<br>=1: New directory every month<br>(by now: **Read only, always=1**)<br>Bit <1>: =1: With single logging, every day a new file will be created.<br>=0: Data will be appended to existing file<br>Bit<2>: With recording by trigger condition:<br>=1: Append to existing file<br>=0: A new file is created, when trigger condition is met<br>Bit<8>: =1: Print date to each line<br>Bit<9>: =1: Print time to each line (**Read only, always=1**)<br>Bit<11>: =1: Print Header. Default=1. Not recommended without header (flag=0), because information will be omitted that may be necessary for later conversion of the file to other formats.<br>**Index 3:** File length lines: Maximum number of rows per file<br>**Index 4:** Decimation factor N: With permanent data recording, if this value is >1, only every Nth measuring value will be logged (value range: 1..65535) |
| out | *value* | Value read, meaning depending on index (see there) |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 222
Device access: Yes, CmdNo: 0x6D
date: 05.10.2017
Applicable devices: GSV-6 with Logger, e.g. GSV-6BT

---

## int CALLTYP GSV86readRawValue (int *ComNo*, int *Chan*)

Read raw measuring value

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Chan* | Input channel to read raw value |

**Returns**

Unscaled raw value in signed int format.

or Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 195

Device access: Yes, CmdNo: 0x3A

date: 20.07.2017
Applicable devices: GSV-8, GSV-6

---

## int CALLTYP GSV86readRTCtime (int *ComNo*, int *ix*, unsigned char * *time*)

Get GSV-6BT/ITA RTC time

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ix* | Index: =0: get actual time. =1: get Alarm time |

| | | |
|---|---|---|
| out | *time* | Array of 6 unsigned char values, where time is written to. time[0]: Year since 2000 (1..99 /255) time[1]: Month of year 1..12 time[2]: Day of Month 1..31 time[3]: Hour 0..23 time[4]: Minute 0..59 time[5]: Seconds 0..59 |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 220
Device access: Yes, CmdNo: 0x6B
date: 05.10.2017
Applicable devices: GSV-6 with RTC, e.g. GSV-6BT

---

## int CALLTYP GSV86readSDfile (int *ComNo*, unsigned char * *data*)

Read content of an open file on SDcard in GSV-6BT/ITA

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *data* | Data read, up to 15 bytes |

**Returns**

: Number of bytes read (0..15) or Simple errorcode: GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Remarks**

: 1. If a whole file is to be read, GSV86copyFromSDfile may be used for convenience

1. File must be open for reading with GSV86openSDfileDir
2. After all data have been read (return value < 15), file must be closed with GSV86controlFileLog(control=0)

OrdinalNo: 228
Device access: Yes, CmdNo: 0x73
date: 05.10.2017
Applicable devices: GSV-6 with Logger, e.g. GSV-6BT

---

## int CALLTYP GSV86readSDfileExt (int *ComNo*, unsigned char * *data*, int *bufSize*)

Read content of an open file on SDcard in GSV-6BT/ITA

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| out | *data* | Data read, up to 270 bytes. Must have been allocated with byte size= bufSize before calling! |
| in | *bufSize* | size of data[] array (used, if < 270). Range: 2..270 |

**Returns**

Number of bytes read (0..270) or Simple errorcode: GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

**Remarks**

1. Permanent data transmission should be stopped (e.g. by using GSV86stopTX) before calling 1st time. Otherwise, only 15 bytes are read and written to data.

2. If a whole file is to be read, GSV86copyFromSDfile may be used instead for convenience
3. File must have been opened for reading with GSV86openSDfileDir
4. After all data have been read - that is, if return value < 270 or < bufSize, respectively - the file must be closed with GSV86controlFileLog(control=0)

OrdinalNo: 241
Device access: Yes, CmdNo: 0x74
date: 11.10.2018

Applicable devices: GSV-6 with Logger, e.g. GSV-6BT

## int CALLTYP GSV86readValueString (int *ComNo*, int *ChanIx*, int *flags*, char * *Vstring*)

Read text string with measuring value(s). (GSV-6, FW-ver >= 3.13)

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ChanIx* | Channel / Index: =0: Read all configured channels, that may be from 1 up to 7 channels. remarks: ChanIx=0 is possible from FW-ver 3.20 on, if:<br>- permanent measuring value transmission is stopped (see GSV86stopTX)<br>flags,bit1 =0 =1..7: Read a single input channel. 7 is always counter / speed channel. An attempt to read an unconfigured channel results in an empty string. No error is thrown in that case. =8: Read temperature of GSV-6 CPU board in °C. =9: Read supply voltage of GSV-6 CPU board in V (available from FW-ver 3.20 on, up to 5V measurable). |
| in | *flags* | Bit 0: =1: Include unit(s) to result string Bit 1: =1: String length fixed to 15 bytes. String may be shortened to fit. Not possible with ChanIx=0. =0: String length variable from 1 to 127 bytes. |
| out | *Vstring* | Resulting string. If Bit 1 of 'flags'=0, at least 127 bytes must have been allocated before. If Bit 1 of 'flags'=1, at least 15 bytes must have been allocated before. |

**Returns**

Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().
OrdinalNo: 242
Device access: Yes, CmdNo: 0x75
date: 21.10.2018
Applicable devices: GSV-6

## int CALLTYP GSV86resetDevice (int *ComNo*)

Reset the GSV-6 / ITA device

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remark: Function sends the command frame, then waits 500ms for the device to boot. It does NOT:

wait for command answer frame (exceptionally, no frame is send by device)
Check if the device is present again / if the communication channel is still open and working!

OrdinalNo: 190

Device access: Yes, CmdNo: 0x78

date: 12.05.2017
Applicable devices: GSV-6

## int CALLTYP GSV86setAutoZeroProperty (int *ComNo*, int *Type*, int *ix*, int *prop*)

Set Properties of the AutoZero function (GSV-8, FWver >= 1.39)

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Type* | Property type to read: =0: ApplyAZ: Channel-flags that specify which channels shall be set to zero, or (with Mode=1) for which channels the AutoZero function is enabled. =1: UseThreshold: With Mode=0: Flags, that specify, which channels shall be compared with their correspondend threshold. The boolan results are then ANDed and if all conditions meet, the SetZero routine will be performed. =2: Mode: Bit 0: =0: Channels are grouped, i.e. if all thershold and time conditions meet (see UseThreshold), the channels configured with ApplyAZ will be set to Zero. Bit 0 =1: Auto-Zero will be applied on all channels configured with ApplyAZ, independently of each other (UseThreshold ignoriered). |
| in | *ix* | reserved (ignored) |
| in | *prop* | Property flags, see description for Type. Type =0 und =1: Channel flags: Bit 0: Condition/Action applied to channel 1, and so forth until Bit 7: applied to channel 8. |

**Returns**

Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().
OrdinalNo: 238
Device access: Yes, CmdNo: 0x99
date: 17.07.2018
Applicable devices: GSV-8

---

## int CALLTYP GSV86setCounterFreqMode (int *ComNo*, int *CntNo*, int *index*, int *mode*)

Write QEI/counter/frequency measuring mode

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *CntNo* | Number of counter (0..1). Must be =0 for GSV-6/ITA. |
| in | *index* | mode to read (see return value) |
| in | *mode* | If **index =0**: General mode / QEI flags. **Bit 0**: =1: Counter module is present **Bit 1**: =1: Counter measuring enabled **Bit 2**: =1: Frequency/Speed measuring enabled. With GSV-6, these two functions can't be used the same time. With GSV-6, they can be used in combination, but for counter No. 0 only. **Bits <4:3>**: QEI mode: 00 FreeRun, non-QEI mode 01 QEI x1 10 QEI x2 11 QEI x4 **Bit 5**: =0: ignore Index/Home input =1: Pulse at Index/Home input copies counter offset to counter value (not recommended with Velocity / frequency measuring) **Bit 6:** =1: Counter saturates to maximum numeric values =0: Counter doesn't saturate (rolls over with sign exchange) **Bit 7**: =0: Device doesn't store counter val in non-volatile memory. At power-up, counter is initialized with 0 =1: At power-down, Device stores actual counter val in non-volatile memory, if Counter measuring enabled **Bit 8**: =1: With Frequency/Speed measuring enabled (Bit2=1), the GSV-6 device uses an additional oszillator for more precise period measuring. GSV-8: Period measuring method. **Bit 9**: =1: Automatic selection of frequency mode: Period measuring or counter (GSV-8 only) **Bit 10**: =0: Pullup-Resistors (10kΩ) **enabled** =1: Pullup- Resistors (10kΩ) **off** (GSV-8 only) |

| | | |
|---|---|---|
| | | **Bit 11**: =1: Index Input inverted (GSV-8 only) Bit 12: =1: Use input noise filter (GSV-8 only)<br>If **index =1**: Frequency / velocity mode: Bits <15:0>: Gate time counter: Counts data frequency periods; if reached, samples counter value and calculates frequency / velocity value. Period measuring: Number of data periods without change, after output value is set to 0.<br>If **index =2**: Counter offset value. Relevant with GeneralMode, bit 5=1 only |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 193

Device access: Yes, CmdNo: 0x6A

date: 22.06.2017

Applicable devices: GSV-8, GSV-6 with counter, e.g. GSV-6BT

---

## int CALLTYP GSV86setLoggerAlarmInterval (int *ComNo*, int *IntervSec*, int *setMode*)

Set interval for slow (power-saving) SD-card Logging. GSV-6BT / ITA

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *IntervSec* | Time interval in seconds for writing a measuring value row to the file |
| in | *setMode* | =0: Switch GSVon/off mode in BGscript to 0 (off!) IntervSec ignored. =1: Set to slow logging with interval in IntervSec from 4 to 59 seconds =2: Set to slow logging with minutes interval. IntervSec: 60..65535 |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 417

Device access: Yes, GSV-6 CPU and BGscript, CmdNo: 0xE0, 0xED, 0xF3, 0xF2, 0xE1, 0xEE, 0x6D, 0x6E

date: 12.01.2018

Applicable devices: GSV-6 with Logger, e.g. GSV-6BT

---

## int CALLTYP GSV86writeAutoZeroSetting (int *ComNo*, int *Type*, int *Chan*, double *val*)

Write Settings of the AutoZero function (GSV-8, FWver >= 1.39). If the measuring value is for a time of <Period> seconds below its correspondent threshold (i.e. around 0 between threshold and -threhold), for certain channels (configurable with SetAutoZeroProperty), a SetZero routine will be performed, if Bit 22 of the Mode-Value is set (see GSV86getMode).

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Type* | Parameter type to read: =0: Write Auto-Zero count period in seconds (Chan ignored) =1: Write Auto-Zero threshold for Chan |
| in | *Chan* | Channel index to write (1..8), if Type=1 |
| in | *val* | Period or threshold |


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

**Returns**

Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().
OrdinalNo: 236
Device access: Yes, CmdNo: 0x97
date: 17.07.2018
Applicable devices: GSV-8

---

## int CALLTYP GSV86writeFTsensorCalArrExt (int *ComNo*, int *Styp*, double * *MatrixB*, int * *Fact1ix*, int * *Fact2ix*, char * *ModelName*)

Set extended calibration values for six-axis sensor calibration (GSV-8, FWver >= 1.39)

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *Styp* | Bits<7:0>: Sensor type: =0: Six-Axis, =1: Six-Axis with MatrixB (2nd order), =2: 3/4 axis sensoor (reserved) Bits<15:8>: Reserved for special calculation type, e.g. Bit 1 (mask 0x100): Way calculation |
| in | *MatrixB* | ptr to array of 36 double values: Calibration matrixB (2nd order) will be written, order: first row 0..5, 2nd row 6..11... Used only if Styp<7:0> =1. |
| in | *Fact1ix* | ptr to array of 6 int values, that are the indices on the raw input value array, used as the 1st factor that form the vector thet the MatrixB is multiplied with. |
| in | *Fact2ix* | ptr to array of 6 int values, that are the indices on the raw input value array, used as the 2nd factor that form the vector thet the MatrixB is multiplied with. |
| in | *ModelName* | ptr to String of maximum of 16 chars that contains the model name of the sensor |

**Returns**

Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText(). Remarks:
1. Values will NOT be stored in non-volatile memory. Therefore, it is adviced to use this function first, and then GSV86writeFTsensorCalArray, which stores the whole sensor calibration structure.

1. For MatrixB, Fact1ix, Fact2ix and ModelName the ptr-values can be NULL. Values are not written in that case. **Pre-condition**: Device password (=User-ID) set (see GSV86setPassword() )

OrdinalNo: 232
Device access: Yes, CmdNo: 0x48, 0x1C
date: 17.07.2018
Applicable devices: GSV-8

---

## int CALLTYP GSV86writeLoggerSettings (int *ComNo*, int *index*, unsigned long *value*)

Write GSV-6BT/ITA file logging settings

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *index* | **Index 0:** Basic Flags:<br>Bit<0>: =1: Suitable SD card inserted<br>Bit<1>: =1: File actually open for recording (read-only)<br>Bit<2>: =1: Recording-enabled-state (persistently stored)<br>Bit<4>: =1: Permanent data recording |

| | | |
|---|---|---|
| | | =0: Single value logging<br>Bit<5>: =1: Recoding by trigger condition (by now, digital input only)<br>**Index 1:** Bit<0>: =1: Alarm Mode: RTC activates alarm-line as configured by alarm interval<br>**Index 2:** Bit<0>: Directory creation mode:<br>Determines, at which time a new directory will be created, when the recording is started:<br>=0: New directory every day<br>=1: New directory every month<br>(by now: **Read only, always=1**)<br>Bit <1>: =1: With single logging, every day a new file will be created.<br>=0: Data will be appended to existing file<br>Bit<2>: With recording by trigger condition:<br>=1: Append to existing file<br>=0: A new file is created, when trigger condition is met<br>Bit<8>: =1: Print date to each line<br>Bit<9>: =1: Print time to each line (**Read only, always=1**)<br>Bit<11>: =1: Print Header. Default=1. Not recommended without header (flag=0), because information will be omitted that may be necessary for later conversion of the file to other formats.<br>**Index 3:** File length lines: Maximum number of rows per file<br>**Index 4:** Decimation factor N: With permanent data recording, if this value is >1, only every Nth measuring value will be logged (value range: 1..65535) |
| in | *value* | Value read, meaning depending on index |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 223
Device access: Yes, CmdNo: 0x6E
date: 05.10.2017
Applicable devices: GSV-6 with Logger, e.g. GSV-6BT

---

## int CALLTYP GSV86writeRTCtime (int *ComNo*, int *ix*, unsigned char * *time*)

Set GSV-6BT/ITA RTC time

**Parameters**

| | | |
|---|---|---|
| in | *ComNo* | Number of Device Comport |
| in | *ix* | Index: =0: set actual time. =1: set Alarm time |
| in | *time* | Array of 6 unsigned char values, where time is read from. IF ix=0 AND time=NULL: SET DECICE RTC TIME FROM CALLERS SYSTEM LOCAL TIME time[0]: Year since 2000 (1..99 /255) time[1]: Month of year 1..12 time[2]: Day of Month 1..31 time[3]: Hour 0..23 time[4]: Minute 0..59 time[5]: Seconds 0..59 |

**Returns**

: Simple errorcode: GSV_OK,if successful or GSV_ERROR if function failed. If GSV_ERROR, more detailed error information ca be retrieved with GSV86getLastProtocollError() or GSV86getLastErrorText().

OrdinalNo: 221

Device access: Yes, CmdNo: 0x6C
date: 05.10.2017
Applicable devices: GSV-6 with RTC, e.g. GSV-6BT

## Data Structures

struct OBJECT_MAPPING

## Macros

```
/*****************************************************************************************
Definitions of constants.
Values can't be changed, since it won't have any effect on using the readily compiled library.
*****************************************************************************************/
/* Version of this DLL */
#define FILEVER_H                       1
#define FILEVER_L                       29

/* Return-codes for Simple errorcode */
#define GSV_OK 0      /*no error, without signalling information */
#define GSV_ERROR -1  /*an error occurred within the request */
#define GSV_TRUE 1    /*no error, with signalling information */

/* Device Model constants */
#define GSV6   6
#define GSV8   8

//#define FEHLSYNC_ERRCNT
/*Constant values used by GSV86*actEx */
#define CONST_BAUDRATE        115200
#define CONST_BUFSIZE 48000

/* Flags for parameter "flags" in GSV86activateExtended */
#define ACTEX_FLAG_HANDSHAKE  4       /* enables HW-handshake in in GSV86activateExtended */
#define ACTEX_FLAG_WAIT_EXTENDED     0x0100  /* waits longer for device-answer in
GSV86activateExtended */
#define ACTEX_FLAG_STOP_TX    0x0200  /* stops continious data transmission in GSV86activateExtended
*/

/* Constants for GSV86getLastProtocollError and GSV86getLastErrorText */
#define ERRTEXT_SIZE 256 /* Maximum size of error text */
#define ERR_MASK_ALL 0xFC000000 /* Mask for error code types */
#define OWN_ERR_MASK 0x20000000 /*Mask, which can be or-ed with one of the well-known Windows Systen
Error Code , see: MSDN:GetLastError() */
#define ERR_MSK_DEVICE 0x38000000 /*Error given by GSV-8 / GSV-6 device, errorcode in Bits<7:0>, see
GSV-8 manual */
/* This application Errorcodes. Retrieve with GSV86getLastProtocollError. See file Errorcodes.h */
```

More Macros documented in MEGSV86w32.h


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

# OBJECT_MAPPING Struct Reference

Struct definition for convenience. ObjMapping parameter can be type-casted to this after return from GSV86getValObjectInfo.
Struct definition for convenience. ObjMapping parameter can be type-casted to this after return from GSV86getValObjectInfo.
Must be packed to total size of 32 Bits, so better use with C or C++ only.

## Data Fields

169 unsigned char ChannelNo
170 unsigned char ValueType
171 unsigned char PhysicType
172 unsigned char reserved

---

## Field Documentation

### unsigned char OBJECT_MAPPING::ChannelNo

Channel-Numbers from 1 to 8

### unsigned char OBJECT_MAPPING::PhysicType

Physical type:
VAL_PHYS_TYPE_NOTDEF =0 no physical type defined (unknown)
VAL_PHYS_TYPE_FORCE_X =1 Force in X-direction
VAL_PHYS_TYPE_FORCE_Y =2 Force in Y-direction
VAL_PHYS_TYPE_FORCE_Z =3 Force in Z-direction
VAL_PHYS_TYPE_TORQUE_X =4 Torque-moment in X-direction
VAL_PHYS_TYPE_TORQUE_Y =5 Torque-moment in Y-direction
VAL_PHYS_TYPE_TORQUE_Z =6 Torque-moment in Z-direction
VAL_PHYS_TYPE_RAW =0x10 Raw value of six-axis sensor (before calculation)
VAL_PHYS_TYPE_TEMP =0x20 Temperature

### unsigned char OBJECT_MAPPING::reserved

(reserved for future use)

### unsigned char OBJECT_MAPPING::ValueType

Value type:
VALTYPE_NORMAL =0 actual measuring value

VALTYPE_MAXVAL =1 Maximum value
VALTYPE_MINVAL =2 Minimum value

---

## valErr Union Reference

### data Fields

173 unsigned char B [6]
174 struct {
175 unsigned short ErrFlags
176 unsigned long Type_Time
177 } err

---

### Field Documentation

### unsigned char valErr::B[6]

### struct { ... } valErr::err

### unsigned short valErr::ErrFlags

### unsigned long valErr::Type_Time

---

```
/* GSV-6 / -8 Device Commands */
#define CMD_RESET_STATUS 0
#define CMD_GETIDENTITY   1
#define CMD_READ_ZERO            2
#define CMD_WRITE_ZERO           3
#define CMD_READOFFSET           4
#define CMD_WRITEOFFSET          5
#define CMD_READ_ANA_SCALE       6
#define CMD_WRITE_ANA_SCALE      7
#define CMD_SET_AOUT_DIRECT      8
#define CMD_GETALL         9
#define CMD_SAVEALL        0x0a
#define CMD_SETZERO 0x0c
#define CMD_GET_AOUT_TYPE 0x0D
#define CMD_SET_AOUT_TYPE 0x0E
#define CMD_GETUNITNO      0x0f
#define CMD_SETUNITNO      0x10
#define CMD_GETUNITTEXT    0x11
#define CMD_SETUNITTEXT    0x12
#define CMD_GETNORM 0x14
#define CMD_SETNORM 0x15
#define CMD_MEGETCAL       0x17
#define CMD_MESETCAL       0x18
#define CMD_MESETID 0x19
#define CMD_MEGETIDSTATE   0x1a
#define CMD_READ_SENSORMODEL     0x1b
#define CMD_WRITE_SENSORMODEL    0x1c
#define CMD_READ_CAL_OPERATOR    0x1d
#define CMD_MEWR_CAL_OPERATOR    0x1e
#define CMD_GETSERNO       0x1f
#define CMD_MESETSERNO     0x20
#define CMD_READ_CAL_DATE  0x21
```


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

```
#define CMD_MEWR_CAL_DATE 0x22
#define CMD_STOP_TRANSMISSION    0x23
#define CMD_START_TRANSMISSION   0x24
#define CMD_CLEARBUFFERABORTTX   0x25
#define CMD_GETMODE 0x26
#define CMD_SETMODE 0x27
#define CMD_GETEQUIPMENT  0x2a
#define CMD_FIRMWAREVERSION      0x2b
#define CMD_MEWRITERANGE  0x34
#define CMD_SETVALPROPS   0x35
#define CMD_GETOPTIONS    0x36
#define CMD_GET_RAW_VALUE 0x3a
#define CMD_GETVALUE      0x3b
#define CMD_CLEARMAXIMUM_VALUE   0x3c
#define CMD_GETLASTERROR  0x42
#define CMD_GET_VALUE_ERROR      0x43
#define CMD_ERASE_ERROR_MEMORY   0x44
#define CMD_GET_SENSOR_PLUGGED   0x45
#define CMD_READSENSORCAL 0x47
#define CMD_WRITESENSORCAL       0x48
#define CMD_GET_TXMAPPING 0x49
#define CMD_SET_TXMAPPING 0x4A
#define CMD_GET_DFILT_TYPE       0x4B
#define CMD_SET_DFILT_TYPE       0x4C
#define CMD_GET_DFILT_CUTOFF     0x4D
#define CMD_SET_DFILT_CUTOFF     0x4E
#define CMD_READ_DFILT_COEF      0x4F
#define CMD_WRITE_DFILT_COEF     0x50
#define CMD_GET_DFILT_ONOFF      0x51
#define CMD_SET_DFILT_ONOFF      0x52
#define CMD_GET_MAXMIN_VAL       0x53
#define CMD_GET_FTSENSOR_ARR_NO  0x54
#define CMD_SET_FTSENSOR_ARR_NO  0x55
#define CMD_GET_DEVICEHOURS      0x56
#define CMD_SET_DEVICEHOURS      0x57
#define CMD_CHANGE_PASSWORD      0x58
#define CMD_GET_DIO_DIRECTION    0x59
#define CMD_SET_DIO_DIRECTION    0x5A
#define CMD_GET_DIO_TYPE  0x5B
#define CMD_SET_DIO_TYPE  0x5C
#define CMD_GET_DIO_LEVEL 0x5D
#define CMD_SET_DIO_LEVEL 0x5E
#define CMD_GET_DO_THRES  0x5F
#define CMD_SET_DO_THRES  0x60
#define CMD_GET_DO_INIT_LEV      0x61
#define CMD_SET_DO_INIT_LEV      0x62
#define CMD_GET_DRATE_RANGE      0x63
#define CMD_GET_TEDS_ENTRY       0x64
#define CMD_READ_TEDS_ARRAY      0x65
#define CMD_WRITE_TEDS_DATA      0x66
#define CMD_STORE_TEDS_DATA      0x67
#define CMD_GET_TEDS_ACTIVE      0x68
#define CMD_GET_COUNTER_MODE     0x69
#define CMD_SET_COUNTER_MODE     0x6A
#define CMD_READ_CLOCK_TIME      0x6B
#define CMD_WRITE_CLOCK_TIME     0x6C
#define CMD_READ_LOG_SETTING     0x6D
#define CMD_WRITE_LOG_SETTING    0x6E
#define CMD_CONTROL_LOGGER       0x6F
#define CMD_QUERY_FILE_SYS       0x70
#define CMD_OPEN_FILE_DIR 0x71
#define CMD_GET_FILEINFO  0x72
#define CMD_READ_FILE            0x73
#define CMD_READ_FILE_EX  0x74
#define CMD_READ_VAL_STRING      0x75
#define CMD_DO_SYSTEM_RESET      0x78
#define CMD_SPECIAL_RESERVED     0x79
#define CMD_RELEASE_INTERFACE    0x7A
#define CMD_GET_INTERF_SETTING   0x7B
```

```
#define CMD_SET_INTERF_SETTING   0x7C
#define CMD_PREPREAD_FTSENSOR    0x7D
#define CMD_STORE_DFILTER 0x7E
#define CMD_STORE_SIXAXCAL       0x7F
#define CMD_GETTXMODE     0x80
#define CMD_SETTXMODE     0x81
#define CMD_READDATARATE  0x8a
#define CMD_WRITEDATARATE 0x8b
#define CMD_GETCANSETTING 0x8c
#define CMD_SETCANSETTING 0x8d
#define CMD_GETANALOGUEFILTER    0x90
#define CMD_SETANALOGUEFILTER    0x91
#define CMD_SWITCHBLOCKING       0x92
#define CMD_GETCOMMANDAVAILABLE  0x93
#define CMD_GET_NOISECUT_THR     0x94
#define CMD_SET_NOISECUT_THR     0x95
#define CMD_READ_AUTOZ_SETTING   0x96
#define CMD_WRITE_AUTOZ_SETTING  0x97
#define CMD_GET_AUTOZ_PROPERTY   0x98
#define CMD_SET_AUTOZ_PROPERTY   0x99
#define CMD_GET_USEROFFSET       0x9A
#define CMD_SET_USEROFFSET       0x9B
#define CMD_GETINPUTTYP          0xA2
#define CMD_SETINPUTTYP          0xA3

#define BTCMD_INIT_CFG_MODE      0xE0
#define BTCMD_EXIT_CFG_MODE      0xE1
#define BTCMD_GET_BT_MAXPOWER    0xE2
#define BTCMD_SET_BT_MAXPOWER    0xE3
#define BTCMD_GET_LE_MAXPOWER    0xE4
#define BTCMD_SET_LE_MAXPOWER    0xE5
#define BTCMD_GET_BTMODE  0xE6
#define BTCMD_SET_BTMODE  0xE7
#define BTCMD_GET_NAME           0xE8
#define BTCMD_SET_NAME           0xE9
#define BTCMD_READ_BAT_VOLT      0xEA
#define BTCMD_DFU_RESET          0xEC
#define BTCMD_GET_GSVONOFF       0xED
#define BTCMD_SET_GSVONOFF       0xEE
#define BTCMD_SET_DEFAULT 0xEF
#define BTCMD_SET_INTYPE  0xEB
#define BTCMD_GET_INTYPE  0xF0
#define BTCMD_SET_DIO            0xF1
#define BTCMD_SET_LOGINTERVAL    0xF2
#define BTCMD_GET_LOGINTERVAL    0xF3
```


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

## Errorcodes.h File Reference

In this file, the error codes are defined. They may have been thrown by the device itself, or by the DLL.
To almost all error codes, two macros are defined for each; in the form
<MACRO_NAME> Error number
<MACRO_NAME>_TXT Error describing text, as returned by GSV86getLastErrorText

### Macros

#### Device Error codes

```
178#define ERR_OK  0x00      /* Ok, no error */
179#define ERR_OK_CHANGED  0x01
180#define ERR_OK_CHANGED_TXT  "Device: No error, but further device parameters
   changed"
181#define ERR_CMD_NOTKNOWN  0x40
182#define ERR_CMD_NOTKNOWN_TXT  "Device: Command number unknown"
183#define ERR_CMD_NOTIMPL  0x41
184#define ERR_CMD_NOTIMPL_TXT  "Device: Command not implemented"
185#define ERR_FRAME_ERROR  0x42
186#define ERR_FRAME_ERROR_TXT  "Device: Frame error: wrong suffix"
187#define ERR_PAR  0x50
188#define ERR_PAR_TXT  "Device: Parameter wrong"
189#define ERR_PAR_ADR  0x51
190#define ERR_PAR_ADR_TXT  "Device: Wrong index or adress parameter"
191#define ERR_PAR_DAT  0x52
192#define ERR_PAR_DAT_TXT  "Device: Wrong data parameter"
193#define ERR_PAR_BITS  0x53
194#define ERR_PAR_BITS_TXT  "Device: Wrong bits inside parameter"
195#define ERR_PAR_ABSBIG  0x54
196#define ERR_PAR_ABSBIG_TXT  "Device: Parameter abolutely too big"
197#define ERR_PAR_ABSMALL  0x55
198#define ERR_PAR_ABSMALL_TXT  "Device: Parameter abolutely too small"
199#define ERR_PAR_COMBI  0x56
200#define ERR_PAR_COMBI_TXT  "Device: Wrong parameter / setting combination"
```

```
#define ERR_PAR_RELBIG  0x57
#define ERR_PAR_RELBIG_TXT  "Device: Parameter too big in relation to other parameters / Settings"
#define ERR_PAR_RELSMALL  0x58
#define ERR_PAR_RELSMALL_TXT  "Device: Parameter too small in relation to other parameters / Settings"
#define ERR_PAR_NOTIMPL  0x59
#define ERR_PAR_NOTIMPL_TXT  "Device: Function invoked by parameter is not implemented"
#define ERR_WRONG_PAR_NUM  0x5B
#define ERR_WRONG_PAR_NUM_TXT  "Device: Wrong number of parameters in frame"
#define ERR_PAR_NOFIT_SETTINGS  0x5C
#define ERR_PAR_NOFIT_SETTINGS_TXT  "Device: Parameter improper with respect to device's settings"
#define ERR_PAR_HW_COLLISION  0x5D
#define ERR_PAR_HW_COLLISION_TXT  "Device: Function leads to hardware (connection) collision, e.g. short-circuit"
#define ERR_NO_DATA_AVAIL  0x60
#define ERR_NO_DATA_AVAIL_TXT  "Device: Data reqested not available (e.g. not initiated)"
#define ERR_DATA_INCONSISTENT  0x61
#define ERR_DATA_INCONSISTENT_TXT  "Device: Data stored not consistent in itself or with parameters"
#define ERR_WRONG_MOD_STATE  0x62
#define ERR_WRONG_MOD_STATE_TXT  "Device: Command could not be executed, because device or functionality in improper state"
#define ERR_NOT_SUPPORTED_D  0x63
#define ERR_NOT_SUPPORTED_D_TXT  "Device: Denied, because requested funcionality not supported"
#define ERR_FDATA_TOO_HIGH  0x64
#define ERR_FDATA_TOO_HIGH_TXT  "Device: Denied, because data rate too high for requested setting"
#define ERR_MEMORY_WRONG_COND  0x6E
#define ERR_MEMORY_WRONG_COND_TXT  "Device: Memory write denied, because condition(s) not satisfied"
#define ERR_MEMORY_ACCESS_DENIED  0x6F
#define ERR_MEMORY_ACCESS_DENIED_TXT  "Device: Memory write: Access denied"
```

```
227 #define ERR_ACC_DEN  0x70
228 #define ERR_ACC_DEN_TXT  "Device: Access denied"
229 #define ERR_ACC_BLK  0x71
230 #define ERR_ACC_BLK_TXT  "Device: Access denied, because write functions are
    blocked"
231 #define ERR_ACC_PWD  0x72
232 #define ERR_ACC_PWD_TXT  "Device: Access denied: Missing password/PIN"
233 #define ERR_ACC_MAXWR  0x74
234 #define ERR_ACC_MAXWR_TXT  "Device: Access denied: Maximum executions reched"
235 #define ERR_ACC_PORT  0x75
236 #define ERR_ACC_PORT_TXT  "Device: Access from this port denied (other port seems
    to have write access)"
237 #define ERR_ACC_RDONLY          0x76
238 #define ERR_ACC_RDONLY_TXT "Device: Write access not allowed"
239 #define ERR_INTERNAL  0x80
240 #define ERR_INTERNAL_TXT  "Internal exception in device. Please contact
    manufacturer"
241 #define ERR_ARITH  0x81
242 #define ERR_ARITH_TXT  "Internal arithmetic exception in device. Please contact
    manufacturer"
243 #define ERR_INTER_ADC  0x82
244 #define ERR_INTER_ADC_TXT  "Device: Erratic behaviour of AD converter. Please
    contact manufacturer"
245 #define ERR_MWERT_ERR  0x83
246 #define ERR_MWERT_ERR_TXT  "Device: Actual measuring value inappropriate to fulfil
    request"
247 #define ERR_EEPROM  0x84
248 #define ERR_EEPROM_TXT  "Device: Erratic behaviour of EEPROM memory. Please
    contact manufacturer"
249 #define ERR_EXT_HW  0x85
250 #define ERR_EXT_HW_TXT "Required external hardware (e.g. SDcard) not present or
    faulty"
251 #define ERR_FILE    0x86
252 #define ERR_FILE_TXT "SDcard: File system driver reports error"
253 #define ERR_WRONG_DIR     0x87
254 #define ERR_WRONG_DIR_TXT "SDcard: Wrong directory / dir. settings inappropriate"
255 #define ERR_RET_TXBUF  0x91
```

```
#define ERR_RET_TXBUF_TXT  "Device transmission buffer full"
#define ERR_RET_BUSY  0x92
#define ERR_RET_BUSY_TXT  "Device too busy to execute request"
#define ERR_RET_RXBUF  0x99
#define ERR_RET_RXBUF_TXT  "Device receive buffer full"

#define GETTEDS_ERR_NOSENSOR  0xB0
#define GETTEDS_ERR_NOSENSOR_TXT  "Device (TEDS): No sensor connected at all"
#define GETTEDS_ERR_NOTEDSEE  0xB1
#define GETTEDS_ERR_NOTEDSEE_TXT  "Device (TEDS): No TEDS memory connected"
#define GETTEDS_ERR_BASICONLY  0xB2
#define GETTEDS_ERR_BASICONLY_TXT  "Device (TEDS): Only Basic TEDS data found"
#define GETTEDS_ERR_NOTEDSDAT  0xB3
#define GETTEDS_ERR_NOTEDSDAT_TXT  "Device (TEDS): Data not in conformance with
  IEEE1541.4"
#define GETTEDS_ERR_ENTRY_INVALID  0xB4
#define GETTEDS_ERR_ENTRY_INVALID_TXT  "Device (TEDS): Data entry not set"
#define GETTEDS_ERR_TOUT  0xB5
#define GETTEDS_ERR_TOUT_TXT  "Device (TEDS): 1-wire EEPROM driver timed out"
#define GETTEDS_ERR_CHKSUM  0xB6
#define GETTEDS_ERR_CHKSUM_TXT  "Device (TEDS): Data checksum error"
#define GETTEDS_ERR_UNKNOWN_TEMPL  0xB7
#define GETTEDS_ERR_UNKNOWN_TEMPL_TXT  "Device (TEDS): TEDS template not
  supported"
#define GETTEDS_ERR_VERIFY_FAIL  0xB8
#define GETTEDS_ERR_VERIFY_FAIL_TXT  "Device (TEDS): Data write-verify failed"
#define BT_CONFIG_ERR 0xC0 //neues BT-Config Interface: Error
#define BT_CONFIG_ERR_TXT "BGscript: BT application error"
```

### Measuring value Error Type

```
#define VALERR_NONE  0
#define VALERR_TYPE_SATURATED  1
#define VALERR_TYPE_MAX_EXCEED  2
#define VALERR_TYPE_SENSOR_BROKEN  3
```

```
285#define ERR_TYPE_ANALOG_OUTPUT  4

286#define ERR_TYPE_DIGITAL_OUTPUT  5
```

### Dll error codes

```
287#define ERR_MUTEXFAILED  0x300000F0    /*d805306608 Mutex request refused by OS */

288#define ERR_MUTEXFAILED_TXT  "MEGSV86xx.DLL: Mutex request refused by OS"

289#define ERR_EVENTFAILED  0x300000F1    /*d805306609 Event request refused by OS */

290#define ERR_EVENTFAILED_TXT  "MEGSV86xx.DLL: Event request refused by OS"

291#define ERR_MEM_ALLOC  0x300000F3      /*d805306611 Memory allocation request
   refused by OS */

292#define ERR_MEM_ALLOC_TXT  "MEGSV86xx.DLL: Memory allocation request refused by
   OS"

293#define ERR_NO_GSV_FOUND  0x300000F4   /*d805306612 Comport could be opened, but
   no GSV answered */

294#define ERR_NO_GSV_FOUND_TXT  "MEGSV86xx.DLL: Com port could be opened, but no GSV
   answered"

295#define ERR_BYTES_WRITTEN  0x300000F5  /*d805306613 Could not write enough bytes
   to the port */

296#define ERR_BYTES_WRITTEN_TXT  "MEGSV86xx.DLL: Could not write enough bytes to the
   port"

297#define ERR_WRONG_PARAMETER  0x30000100      /*d805306624 Function parameter
   exceedance */

298#define ERR_WRONG_PARAMETER_TXT  "MEGSV86xx.DLL: Function parameter exceedance"

299#define ERR_NO_GSV_ANSWER  0x30000058  /*d805306456 Command response from device
   timed out */

300#define ERR_NO_GSV_ANSWER_TXT  "MEGSV86xx.DLL: Command response from device timed
   out"

301#define ERR_WRONG_ANSWER_NUM  0x30000059     /*d805306457 Parameter number in
   command answer frame not as expected */

302#define ERR_WRONG_ANSWER_NUM_TXT  "MEGSV86xx.DLL: Parameter number in command
   answer frame not as expected"

303#define ERR_WRONG_ANSWER  0x30000060   /*d805306458 GSV-8 sended wrong command
   answer */

304#define ERR_WRONG_ANSWER_TXT  "MEGSV86xx.DLL: GSV-8 sended wrong command answer"

305#define ERR_WRONG_FRAME_SUFFIX  0x30000061 /*d805306465 Frame suffix wrong */

306#define ERR_WRONG_FRAME_SUFFIX_TXT  "MEGSV86xx.DLL: Frame suffix from device
   wrong"

307#define ERR_NOT_SUPPORTED  0x30000062  /*GSV-8 Firmware doesn't support the
   request */
```

```
#define ERR_NOT_SUPPORTED_TXT  "MEGSV86xx.DLL: Device firmware doesn't support the request"

#define ERR_WRONG_COMNO  0x30000101    /*d805306625 ComNo parameter wrong */

#define ERR_WRONG_COMNO_TXT  "MEGSV86xx.DLL: ComNo parameter wrong"

#define ERR_COM_ALREADY_OPEN  0x300000F6      /*d805306614 Comport requested for opening (GSV86activateExtended) is already open */

#define ERR_COM_ALREADY_OPEN_TXT  "MEGSV86xx.DLL: Comport requested for opening is already open"

#define ERR_COM_GEN_FAILURE  0x3000001F /*hardware-or driver-error of COMport. See: ERROR_GEN_FAILURE @msdn.microsoft.com/en-us/library/ms681382(VS.85).aspx */

#define ERR_COM_GEN_FAILURE_TXT  "MEGSV86xx.DLL: Hardware-or driver-error of COMport (Generic error: System Error 0x1F)"

#define ERR_INTERNAL_FUNC  0x30000105  /*internal function call failed */

#define ERR_INTERNAL_FUNC_TXT  "MEGSV86xx.DLL: Internal function call failed"

#define ERR_PARAM_NOT_STORED  0x30000065      /*Parameter is not stored (correctly) in the device memory */

#define ERR_PARAM_NOT_STORED_TXT  "MEGSV86xx.DLL: Parameter is not stored (correctly) in the device memory"

#define ERR_FILE_CONTENT  0x30000108   /* File passed by user has error in content */

#define ERR_FILE_CONTENT_TXT  "MEGSV86xx.DLL: File passed by user has error in its content"

#define ERR_UNKNOWN_VALUE  0x3000010A  /* Value read from device can not be deocded */

#define ERR_UNKNOWN_VALUE_TXT  "MEGSV86xx.DLL: Value read from device can not be deocded"

#define ERR_WRONG_COMM_STATE 0x300000C1

#define ERR_WRONG_COMM_STATE_TXT "MEGSV86xx.DLL: Communication module in improper state"

#define ERR_WAIT_CMD_RESSOURCE_TOUT 0x30000069

#define ERR_WAIT_CMD_RESSOURCE_TOUT_TXT "MEGSV86xx.DLL: Waiting for I/O access ressource timed out"

#define ERR_DLL_READTHREAD 0x300000F8

#define ERR_DLL_READTHREAD_TXT "MEGSV86xx.DLL: Own I/O thread invalid"
```

### Digital filter dll functions error codes

```
#define NO_ERR  0

#define DF_ERR_MASK  0x31000000

#define DF_ERR_NOT_INIT  0x30000201

#define DF_ERR_NOT_INIT_TXT  "DLL.Dfilter: Digital filter function could not be executed, because digital filter not initialized"
```

```
333#define DF_ERR_OPT_WRONG  0x30000202
334#define DF_ERR_OPT_WRONG_TXT  "DLL.Dfilter: Wrong or incompatible filter options"
335#define DF_ERR_NO_CONVERGENCE  0x30000203
336#define DF_ERR_NO_CONVERGENCE_TXT  "DLL.Dfilter: Digital filter calculation failed
   to converge"
337#define DF_ERR_CHEB_RIB_WRONG  0x30000204
338#define DF_ERR_CHEB_BUG_Y  0x30000205
339#define DF_ERR_ZERO_NUM_HIGH  0x30000206
340#define DF_ERR_POLE_ZERO_NOT_CONJ  0x30000207
341#define DF_ERR_COEFF_SUM_TOOBIG  0x30000208
342#define DF_ERR_COEFF_SUM_TOOBIG_TXT  "DLL.Dfilter: Resulting coefficients
   discarded, because they may limit measuring precision (sum too big)"
343#define DF_ERR_INTERN_GAIN_TOO_BIG  0x30000209
344#define DF_ERR_INTERN_GAIN_TOO_BIG_TXT  "DLL.Dfilter: Resulting coefficients
   discarded, because they may limit measuring precision (gain too big)"
345#define DF_ERR_FIR_ODD_ORDER_NOTALLOWED  0x3000020A
346#define DF_ERR_FIR_ODD_ORDER_NOTALLOWED_TXT  "DLL.Dfilter: Odd filter order not
   allowed with this release"
```

## Macro Definition Documentation

```
#define DF_ERR_CHEB_BUG_Y  0x30000205
#define DF_ERR_CHEB_RIB_WRONG  0x30000204
#define DF_ERR_COEFF_SUM_TOOBIG  0x30000208
#define DF_ERR_COEFF_SUM_TOOBIG_TXT  "DLL.Dfilter: Resulting coefficients discarded,
because they may limit measuring precision (sum too big)"
#define DF_ERR_FIR_ODD_ORDER_NOTALLOWED  0x3000020A
#define DF_ERR_FIR_ODD_ORDER_NOTALLOWED_TXT  "DLL.Dfilter: Odd filter order not allowed
with this release"
#define DF_ERR_INTERN_GAIN_TOO_BIG  0x30000209
#define DF_ERR_INTERN_GAIN_TOO_BIG_TXT  "DLL.Dfilter: Resulting coefficients discarded,
because they may limit measuring precision (gain too big)"
#define DF_ERR_MASK  0x31000000
#define DF_ERR_NO_CONVERGENCE  0x30000203
#define DF_ERR_NO_CONVERGENCE_TXT  "DLL.Dfilter: Digital filter calculation failed to
converge"
#define DF_ERR_NOT_INIT  0x30000201
#define DF_ERR_NOT_INIT_TXT  "DLL.Dfilter: Digital filter function could not be executed,
because digital filter not initialized"
#define DF_ERR_OPT_WRONG  0x30000202
#define DF_ERR_OPT_WRONG_TXT  "DLL.Dfilter: Wrong or incompatible filter options"
```

```
#define DF_ERR_POLE_ZERO_NOT_CONJ  0x30000207
#define DF_ERR_ZERO_NUM_HIGH  0x30000206
#define ERR_ACC_BLK  0x71
#define ERR_ACC_BLK_TXT  "Device: Access denied, because write functions are blocked"
#define ERR_ACC_DEN  0x70
#define ERR_ACC_DEN_TXT  "Device: Access denied"
#define ERR_ACC_MAXWR  0x74
#define ERR_ACC_MAXWR_TXT  "Device: Access denied: Maximum executions reched"
#define ERR_ACC_PORT  0x75
#define ERR_ACC_PORT_TXT  "Device: Access from this port denied (other port seems to have
write access)"
#define ERR_ACC_RDONLY           0x76
#define ERR_ACC_RDONLY_TXT "Device: Write access not allowed"
#define ERR_ACC_PWD  0x72
#define ERR_ACC_PWD_TXT  "Device: Access denied: Missing password/PIN"
#define ERR_ARITH  0x81
#define ERR_ARITH_TXT  "Internal arithmetic exception in device. Please contact
manufacturer"
#define ERR_BYTES_WRITTEN  0x300000F5  /*d805306613 Could not write enough bytes to the
port */
#define ERR_BYTES_WRITTEN_TXT  "MEGSV86xx.DLL: Could not write enough bytes to the port"
#define ERR_CMD_NOTIMPL  0x41
#define ERR_CMD_NOTIMPL_TXT  "Device: Command not implemented"
#define ERR_CMD_NOTKNOWN  0x40
#define ERR_CMD_NOTKNOWN_TXT  "Device: Command number unknown"
#define ERR_COM_ALREADY_OPEN  0x300000F6      /*d805306614 Comport requested for opening
(GSV86activateExtended) is already open */
#define ERR_COM_ALREADY_OPEN_TXT  "MEGSV86xx.DLL: Comport requested for opening is
already open"
#define ERR_COM_GEN_FAILURE  0x3000001F /*hardware-or driver-error of COMport. See:
ERROR_GEN_FAILURE @msdn.microsoft.com/en-us/library/ms681382(VS.85).aspx */
#define ERR_COM_GEN_FAILURE_TXT  "MEGSV86xx.DLL: Hardware-or driver-error of COMport
(Generic error: System Error 0x1F)"
#define ERR_DATA_INCONSISTENT  0x61
#define ERR_DATA_INCONSISTENT_TXT  "Device: Data stored not consistent in itself or with
parameters"
#define ERR_EEPROM  0x84
#define ERR_EEPROM_TXT  "Device: Erratic behaviour of EEPROM memory. Please contact
manufacturer"
#define ERR_EVENTFAILED  0x300000F1    /*d805306609 Event request refused by OS */
#define ERR_EVENTFAILED_TXT  "MEGSV86xx.DLL: Event request refused by OS"
#define ERR_FDATA_TOO_HIGH  0x64
#define ERR_FDATA_TOO_HIGH_TXT  "Device: Denied, because data rate too high for requested
setting"
#define ERR_FILE_CONTENT  0x30000108   /* File passed by user has error in content */
#define ERR_FILE_CONTENT_TXT  "MEGSV86xx.DLL: File passed by user has error in its
content"
```

```
#define ERR_FRAME_ERROR  0x42

#define ERR_FRAME_ERROR_TXT  "Device: Frame error: wrong suffix"

#define ERR_INTER_ADC  0x82

#define ERR_INTER_ADC_TXT  "Device: Erratic behaviour of AD converter. Please contact
manufacturer"

#define ERR_INTERNAL  0x80

#define ERR_INTERNAL_FUNC  0x30000105  /*internal function call failed */

#define ERR_INTERNAL_FUNC_TXT  "MEGSV86xx.DLL: Internal function call failed"

#define ERR_INTERNAL_TXT  "Internal exception in device. Please contact manufacturer"

#define ERR_MEM_ALLOC  0x300000F3      /*d805306611 Memory allocation request refused by
OS */

#define ERR_MEM_ALLOC_TXT  "MEGSV86xx.DLL: Memory allocation request refused by OS"

#define ERR_MEMORY_ACCESS_DENIED  0x6F

#define ERR_MEMORY_ACCESS_DENIED_TXT  "Device: Memory write: Access denied"

#define ERR_MEMORY_WRONG_COND  0x6E

#define ERR_MEMORY_WRONG_COND_TXT  "Device: Memory write denied, because condition(s) not
satisfied"

#define ERR_MUTEXFAILED  0x300000F0    /*d805306608 Mutex request refused by OS */

#define ERR_MUTEXFAILED_TXT  "MEGSV86xx.DLL: Mutex request refused by OS"

#define ERR_MWERT_ERR  0x83

#define ERR_MWERT_ERR_TXT  "Device: Actual measuring value inappropriate to fulfil
request"

#define ERR_NO_DATA_AVAIL  0x60

#define ERR_NO_DATA_AVAIL_TXT  "Device: Data reqested not available (e.g. not initiated)"

#define ERR_NO_GSV_ANSWER  0x30000058  /*d805306456 Command response from device timed
out */

#define ERR_NO_GSV_ANSWER_TXT  "MEGSV86xx.DLL: Command response from device timed out"

#define ERR_NO_GSV_FOUND  0x300000F4   /*d805306612 Comport could be opened, but no GSV
answered */

#define ERR_NO_GSV_FOUND_TXT  "MEGSV86xx.DLL: Com port could be opened, but no GSV
answered"

#define ERR_NOT_SUPPORTED  0x30000062  /*GSV-8 Firmware doesn't support the request */

#define ERR_NOT_SUPPORTED_D  0x63

#define ERR_NOT_SUPPORTED_D_TXT  "Device: Denied, because requested funcionality not
supported"

#define ERR_NOT_SUPPORTED_TXT  "MEGSV86xx.DLL: Device firmware doesn't support the
request"

#define ERR_OK  0x00      /* Ok, no error */

#define ERR_OK_CHANGED  0x01

#define ERR_OK_CHANGED_TXT  "Device: No error, but further device parameters changed"

#define ERR_PAR  0x50

#define ERR_PAR_ABSBIG  0x54

#define ERR_PAR_ABSBIG_TXT  "Device: Parameter abolutely too big"

#define ERR_PAR_ABSMALL  0x55

#define ERR_PAR_ABSMALL_TXT  "Device: Parameter abolutely too small"

#define ERR_PAR_ADR  0x51
```

```
#define ERR_PAR_ADR_TXT  "Device: Wrong index or adress parameter"

#define ERR_PAR_BITS  0x53

#define ERR_PAR_BITS_TXT  "Device: Wrong bits inside parameter"

#define ERR_PAR_COMBI  0x56

#define ERR_PAR_COMBI_TXT  "Device: Wrong parameter / setting combination"

#define ERR_PAR_DAT  0x52

#define ERR_PAR_DAT_TXT  "Device: Wrong data parameter"

#define ERR_PAR_HW_COLLISION  0x5D

#define ERR_PAR_HW_COLLISION_TXT  "Device: Function leads to hardware (connection)
collision, e.g. short-circuit"

#define ERR_PAR_NOFIT_SETTINGS  0x5C
#define ERR_PAR_NOFIT_SETTINGS_TXT  "Device: Parameter improper with respect to device's
settings"

#define ERR_PAR_NOTIMPL  0x59
#define ERR_PAR_NOTIMPL_TXT  "Device: Function invoked by parameter is not implemented"

#define ERR_PAR_RELBIG  0x57
#define ERR_PAR_RELBIG_TXT  "Device: Parameter too big in relation to other parameters /
Settings"

#define ERR_PAR_RELSMALL  0x58

#define ERR_PAR_RELSMALL_TXT  "Device: Parameter too small in relation to other
parameters / Settings"

#define ERR_PAR_TXT  "Device: Parameter wrong"

#define ERR_PARAM_NOT_STORED  0x30000065      /*Parameter is not stored (correctly) in
the device memory */

#define ERR_PARAM_NOT_STORED_TXT  "MEGSV86xx.DLL: Parameter is not stored (correctly) in
the device memory"

#define ERR_RET_BUSY  0x92

#define ERR_RET_BUSY_TXT  "Device too busy to execute request"

#define ERR_RET_RXBUF  0x99

#define ERR_RET_RXBUF_TXT  "Device receive buffer full"

#define ERR_RET_TXBUF  0x91

#define ERR_RET_TXBUF_TXT  "Device transmission buffer full"

#define ERR_TYPE_ANALOG_OUTPUT  4

#define ERR_TYPE_DIGITAL_OUTPUT  5

#define ERR_UNKNOWN_VALUE  0x3000010A  /* Value read from device can not be deocded */

#define ERR_UNKNOWN_VALUE_TXT  "MEGSV86xx.DLL: Value read from device can not be deocded"

#define ERR_WRONG_ANSWER  0x30000060    /*d805306458 GSV-8 sended wrong command answer */
#define ERR_WRONG_ANSWER_NUM  0x30000059      /*d805306457 Parameter number in command
answer frame not as expected */
#define ERR_WRONG_ANSWER_NUM_TXT  "MEGSV86xx.DLL: Parameter number in command answer
frame not as expected"
#define ERR_WRONG_ANSWER_TXT  "MEGSV86xx.DLL: GSV-8 sended wrong command answer"
#define ERR_WRONG_COMNO  0x30000101     /*d805306625 ComNo parameter wrong */
#define ERR_WRONG_COMNO_TXT  "MEGSV86xx.DLL: ComNo parameter wrong"
#define ERR_WRONG_FRAME_SUFFIX  0x30000061 /*d805306465 Frame suffix wrong */
#define ERR_WRONG_FRAME_SUFFIX_TXT  "MEGSV86xx.DLL: Frame suffix from device wrong"

#define ERR_WRONG_MOD_STATE  0x62

#define ERR_WRONG_MOD_STATE_TXT  "Device: Command could not be executed, because device
or functionality in improper state"

#define ERR_WRONG_PAR_NUM  0x5B
```

```
#define ERR_WRONG_PAR_NUM_TXT  "Device: Wrong number of parameters in frame"

#define ERR_WRONG_PARAMETER  0x30000100       /*d805306624 Function parameter exceedance
*/
#define ERR_WRONG_PARAMETER_TXT  "MEGSV86xx.DLL: Function parameter exceedance"
#define GETTEDS_ERR_BASICONLY  0xB2
#define GETTEDS_ERR_BASICONLY_TXT  "Device (TEDS): Only Basic TEDS data found"
#define GETTEDS_ERR_CHKSUM  0xB6
#define GETTEDS_ERR_CHKSUM_TXT  "Device (TEDS): Data checksum error"
#define GETTEDS_ERR_ENTRY_INVALID  0xB4
#define GETTEDS_ERR_ENTRY_INVALID_TXT  "Device (TEDS): Data entry not set"
#define GETTEDS_ERR_NOSENSOR  0xB0
#define GETTEDS_ERR_NOSENSOR_TXT  "Device (TEDS): No sensor connected at all"
#define GETTEDS_ERR_NOTEDSDAT  0xB3
#define GETTEDS_ERR_NOTEDSDAT_TXT  "Device (TEDS): Data not in conformance with
IEEE1541.4"
#define GETTEDS_ERR_NOTEDSEE  0xB1
#define GETTEDS_ERR_NOTEDSEE_TXT  "Device (TEDS): No TEDS memory connected"
#define GETTEDS_ERR_TOUT  0xB5
#define GETTEDS_ERR_TOUT_TXT  "Device (TEDS): 1-wire EEPROM driver timed out"
#define GETTEDS_ERR_UNKNOWN_TEMPL  0xB7
#define GETTEDS_ERR_UNKNOWN_TEMPL_TXT  "Device (TEDS): TEDS template not supported"
#define GETTEDS_ERR_VERIFY_FAIL  0xB8
#define GETTEDS_ERR_VERIFY_FAIL_TXT  "Device (TEDS): Data write-verify failed"
#define NO_ERR  0
#define VALERR_NONE  0
#define VALERR_TYPE_MAX_EXCEED  2
#define VALERR_TYPE_SATURATED  1

#define VALERR_TYPE_SENSOR_BROKEN  3
```

# Annex

## Digital-I/O Numbers

In the devices and windows API (DLL), the numbers of the DIOs are assigned to the terminal connection identification as follows:

| Number in the API and terminal program | Belongs to group | Identification on the terminal board |
|---|---|---|
| 1 | 1 | 1.1 |
| 2 | 1 | 1.2 |
| 3 | 1 | 1.3 |
| 4 | 1 | 1.4 |
| 5 | 2 | 2.1 |
| 6 | 2 | 2.2 |
| 7 | 2 | 2.3 |
| 8 | 2 | 2.4 |
| 9 | 3 | 3.1 |
| 10 | 3 | 3.2 |
| 11 | 3 | 3.3 |
| 12 | 3 | 3.4 |
| 13 | 4 | 4.1 |
| 14 | 4 | 4.2 |
| 15 | 4 | 4.3 |
| 16 | 4 | 4.4 |

## Digital I/O Functions

The following functions can be configured:

| No | Function | Data direction | Parameter Device- or DLL-Command GSV86Get/ SetDIOtype | Short description |
|---|---|---|---|---|
| 0 | None | n.a. | 0x000000 | IO not present (GSV-6). Read-only |
| 1 | General-Purpose Input | Input | 0x000004 | General input. The logic level can be queried with GetDIOlevel / GSV86getDIOlevel. |


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

| | | | | |
|---|---|---|---|---|
| 2 | Zero setting single channel | Input | 0x000010 | The active input level sets an analogue input channel to zero. |
| 3 | Zero setting all channels | Input | 0x000020 | The active input level sets all analogue input channels to zero. |
| 4 | Resetting the maximum and minimum value determination | Input | 0x000040 | The active input level resets all maximum and minimum values. |
| 5 | Resetting digital outputs to default levels | Input | 0x000050 | The active input level sets all digital IOs configured as outputs to their configured default levels |
| 6 | Trigger send actual value | Input | 0x000080 | Triggers the sending of a measured value frame with actual measured values via a USB interface to the inactive-to-active edge of the digital input. |
| 7 | Trigger maximum value | Input | 0x000100 | The maximum value determination is started for the inactive-to-active edge at the digital input (all input channels) and a frame with these maximum values is sent to the USB interface at the active-to-inactive edge. |
| 8 | Trigger minimum value | Input | 0x000200 | The minimum value determination is started for the inactive-to-active edge at the digital input (all input channels) and a frame with these minimum values is sent to the USB interface at the active-to-inactive edge. |
| 9 | Trigger mean value | Input | 0x000400 | A decimating mean value formation is started for the inactive-to-active edge on the digital input (all input channels) and a frame with these mean values is sent to the USB interface at the active-to-inactive edge. |
| 10 | Trigger send actual value | Input | 0x000800 | While the input level is active, measured value frames with actual measured values are sent via a USB interface at the set data rate. |
| 11 | Sync Slave | Input | 0x000002 | At low-to-high edge at digital input, a measuring value frame is transmitted. Only if used with a second GSV-8, configured as Sync Master, and whose sync output (see No.19) is wired to the Sync Input of the Slave. |
| 12 | QEI-Encoder | Input | 0x000008 | Input for quadrature-encoder / counter frequency. Read-Only. To change, the command Write Counter/Freq Mode can be |

| | | | | |
|---|---|---|---|---|
| | | | | used at index 0. |
| 13 | File-Logging | Input | 0x000001 | Trigger input for recording measuring data in a file. GSV-6 with SD-card / Logger only. |
| 14 | General purpose output | Output | 0x001000 | General output. The actual logic level can be defined with SetDIOlevel / GSV86setDIOlevel. |
| 15 | Threshold output actual measured values | Output | 0x010000 | Threshold value output: The output is activated if the assigned measured value is larger than the upper threshold value and is deactivated if it is smaller than the lower threshold value. |
| 16 | Threshold output Maximum value | Output | 0x014000 | Threshold value output: The output is activated if the assigned maximum value is larger than the upper threshold value and is deactivated if it is smaller than the lower threshold value. |
| 17 | Threshold output minimum value | Output | 0x018000 | Threshold value output: The output is activated if the assigned minimum value is larger than the upper threshold value and is deactivated if it is smaller than the lower threshold value. |
| 18 | Window comparator output actual measured value | Output | 0x012000 | Window comparator: The output is activated if the assigned measured value is smaller than the upper threshold value and larger than the lower threshold value; otherwise it is deactivated. |
| 19 | Window comparator output maximum value | Output | 0x016000 | Window comparator: The output is activated if the assigned maximum value is smaller than the upper threshold value and larger than the lower threshold value; otherwise it is deactivated. |
| 20 | Window comparator output minimum value | Output | 0x01A000 | Window comparator: The output is activated if the assigned minimum value is smaller than the upper threshold value and larger than the lower threshold value; otherwise it is deactivated. |
| 21 | Sync-Master | Output | 0x020000 | The Sync Master generates a synchronization signal synchronously to its measuring frame transmission. The Slaves are wired to this digital sync output (see No. 11). When transmission is started, the level edge is low-to-high, at the half of the data period, it's high-to-low. |

### Inverting digital inputs

The DIOs have pull-up resistances that generate high levels when the input is open. For input trigger functions that are intended to be used with a switch or button, that one must be connected between the DIO and the GNDD terminal. The line must be functionally inverted by software so that the function can be executed when the switch is closed. When


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

using the device interfaces or DLL, the specified value in the above mentioned column 'Value' must be ORed with 0x80000 for this purpose.

The threshold value outputs can also be inverted in this way.

The terms in the above mentioned table mean:

| Level | Non-inverted | Inverted |
|---|---|---|
| Active | Logic 1 = High = 5V | Logic 0 = Low = 0V |
| Inactive | Logic 0 = Low = 0V | Logic 1 = High = 5V |

Only with the General-purpose- and the Master-slave-sync functions (No. 1,11, 13 and 20 in the table), inverting has no effect. The functions GSV86get/setDIOlevel and Get/SetDIOlevel always read the level directly, i.e. not inverted.

**Other Notes Digital I/O**

The default level can be defined for digital outputs, i.e. the level that the output should take after restarting and after a reconfiguration. This setting also applies directly, i.e. independent of the inversion state.

The general constant data transmission should be turned off for measured value-send-trigger functions (no. 6 to 10 in the above-mentioned table). This can be done with the button y in the terminal program.

For functions, that are associated with the acquisition of maximum and minimum values (in the above-mentioned table no. 4,7,8,16,17,19,20) the determination of maximum and minimum values of the firmware should be activated. This can be done with the button m in the terminal program.

The threshold switch functions react on the absolute values, if the type is ORed with 0x400000, i.e. the thresholds are 'mirrored' into the negative range and both polarities of measuring values can trigger the threshold switch.[1]

Further notes can be found in the general manual, chapter "Digital I/O Functions".

1 As by Jan. 2020: Reserved, coming feature.

## Unit Numbers

| | | | |
|---|---|---|---|
| 0: | mV/V | 31: | Pa |
| 1: | kg | 32: | hPa |
| 2: | g | 33: | MPa |
| 3: | N | 34: | N/mm² |
| 4: | cN | 35: | ° |
| 5: | V | 36: | Hz |
| 6: | µm/m | 37: | m/s |
| 7: | (none) | 38: | km/h |
| 8: | t | 39: | m³/h |
| 9: | kN | 40: | mA |
| 10: | lb | 41: | A |
| 11: | oz | 42: | m/s² |
| 12: | kp | 43: | flbs |
| 13: | lbf | 44: | ftlb |
| 14: | pdl | 45: | J |
| 15: | mm | 46: | kWh |
| 16: | m | | |
| 17: | cNm | | |
| 18: | Nm | | |
| 19: | °C | | |
| 20: | °F | | |
| 21: | K | | |
| 22: | oztr | | |
| 23: | dwt | | |
| 24: | kNm | | |
| 25: | % | | |
| 26: | 0/00 | | |
| 27: | W | | |
| 28: | kW | | |
| 29: | rpm | | |
| 30: | bar | | |


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

## Error codes for GSV86getValueError with GSV-8

| Error-Type: Type of fault (category) | | |
|---|---|---|
| Value | Name | Meaning |
| 1 | VALERR_TYPE_SATURATED | Value at sensor input saturated, i.e. input measuring range exceeded |
| 2 | VALERR_TYPE_MAX_EXCEED | Allowed maximum physical value exceeded (especially for Multi-axis sensor) |
| 3 | VALERR_TYPE_SENSOR_BROKEN | Bridge sensor or its cable damaged |
| 4 | HWERR_TYPE_ANA_OUT | Analog output configured as current output is unconnected or output driver overheated |
| 5 | HWERR_TYPE_DIO | Digital I/O line configured as output is shorted (wrong level) |

**Return value:** ***ErrInfo*** Error-Flags: Type-dependently coded.

Description:

### 1. ErrorType = VALERR_TYPE_SATURATED:

Bits<15:8>: If Bit=1: Negative saturation occurred in one or several input channels, whereby, Bit 8 corresponds to channel 1, Bit 9 channel 2, and so on, to Bit 15: channel 8.

Bits<7:0>: If Bit=1: Positive saturation occurred in one or several input channels, whereby, Bit 8 corresponds to channel 1, Bit 9 channel 2, and so on, to Bit 15: channel 8.

**Remarks**:

If the saturation is actually present, bit 0 of the *ErrFlags* parameter of GSV86readMultiple is also set. In that condition, the red "FUNCTION"-LED is lit permanently.

### 2. ErrType = VALERR_TYPE_MAX_EXCEED:

#### 2.1. With Force-Torque (Six-axis) sensor

Bit 0: If =1: In the **Fx** direction an (positive or negative) exceedance of the maximum value defined in the six-axis-sensors calibration data occurred.

Bit 1: If =1: In the **Fy** direction an (positive or negative) exceedance of the maximum value defined in the six-axis-sensors calibration data occurred.

Bit 2: If =1: In the **Fz** direction an (positive or negative) exceedance of the maximum value defined in the six-axis-sensors calibration data occurred.

Bit 3: If =1: In the **Mx** direction an (positive or negative) exceedance of the maximum value defined in the six-axis-sensors calibration data occurred.

Bit 4: If =1: In the **My** direction an (positive or negative) exceedance of the maximum value defined in the six-axis-sensors calibration data occurred.

Bit 5: If =1: In the **Mz** direction an (positive or negative) exceedance of the maximum value defined in the six-axis-sensors calibration data occurred.

### 2.2. With PT1000 Temperature sensor: Bits<7:0> = <15:8>

Bits<7:0> At input channel (BitNo+1), a maximum exceedance occurred. The measuring value is set to 9999.0 at positive exceedance and -9999.0 at negative exceedance. To distinct from Force-Torque error, the corresponding Bits in the high byte (bits<15:8>) is also set, whereby BitNo = BitNo+8.

**Remarks**:

If the range exceedance is actually present, bit 1 of the *ErrFlags* parameter of GSV86readMultiple is also set. In that condition, the red "FUNCTION"-LED is lit permanently.

## 3. ErrType = VALERR_TYPE_SENSOR_BROKEN:

| | |
|---|---|
| Bit 0: | If =1: fault condition at Ud+ line of input channel 1 |
| Bit 1: | If =1: fault condition at Ud- line of input channel 1 |
| Bit 2: | If =1: fault condition at Ud+ line of input channel 2 |
| Bit 3: | If =1: fault condition at Ud- line of input channel 2 |
| Bit 4: | If =1: fault condition at Ud+ line of input channel 3 |
| Bit 5: | If =1: fault condition at Ud- line of input channel 3 |
| Bit 6: | If =1: fault condition at Ud+ line of input channel 4 |
| Bit 7: | If =1: fault condition at Ud- line of input channel 4 |
| Bit 8: | If =1: fault condition at Ud+ line of input channel 5 |
| Bit 9: | If =1: fault condition at Ud- line of input channel 5 |
| Bit 10: | If =1: fault condition at Ud+ line of input channel 6 |
| Bit 11: | If =1: fault condition at Ud- line of input channel 6 |
| Bit 12: | If =1: fault condition at Ud+ line of input channel 7 |
| Bit 12: | If =1: fault condition at Ud- line of input channel 7 |
| Bit 14: | If =1: fault condition at Ud+ line of input channel 8 |
| Bit 15: | If =1: fault condition at Ud- line of input channel 8 |

## 4. ErrType = HWERR_TYPE_ANA_OUT:

Constant value 0xFFFF: Fault in any of the analog output channels. Else:

| | |
|---|---|
| Bit 0: | If =1: Open current output at channel 1 |
| Bit 1: | If =1: Open current output at channel 2 |
| Bit 2: | If =1: Open current output at channel 3 |
| Bit 3: | If =1: Open current output at channel 4 |
| Bit 4: | If =1: Open current output at channel 5 |
| Bit 5: | If =1: Open current output at channel 6 |
| Bit 6: | If =1: Open current output at channel 7 |
| Bit 7: | If =1: Open current output at channel 8 |
| Bit 8: | If =1: Output driver overheated at channel 1 |


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

Bit 9: If =1: Output driver overheated at channel 2

Bit 10: If =1: Output driver overheated at channel 3

Bit 11: If =1: Output driver overheated at channel 4

Bit 12: If =1: Output driver overheated at channel 5

Bit 13: If =1: Output driver overheated at channel 6

Bit 14: If =1: Output driver overheated at channel 7

Bit 15: If =1: Output driver overheated at channel 8

**Remark**:

The reason for an overheated output driver may be a short-circuit at a voltage output.

### 5. ErrType = HWERR_TYPE_DIO:

Bits<15:0> If Bit=1: A short circuit occurred at the corresponding digital output, i.e. If configured as an output and switched to high, it may be shorted to GND, or if it's switched to low, a voltage >=3V may be applied.

Bit 0 corresponds to DIOno 1 (Group1: 1.1.), Bit 1 DIOno 2 (Group1: 1.2.), and so forth, to Bit 15: DIOno 16 (Group4: 4.4.)

### Return value *ErrInfo* with GSV-6:

**Bits 5:0:** Exceedance of the maximum value defined in the six-axis-sensors calibration data occurred at channel(s) defined by bit No: Bit 0: Channel 1, bit 1: Channel 2 ... bit 5: Ch6

**Bits 13:8:** Saturation of input channel(s) occurred at channel(s) defined by bit No: Bit 8: Channel 1, bit 9: Channel 2 ... bit 13: Ch. 6

## Flags and Enumeratios for  Read / Write Interface Settings

### Basic Settings: Physical Interface type

| Enum-No. | Meaning |
|---|---|
| 1 | RS232 Interface (asynchronous, 8N1) V24 levels |
| 2 | RS232 Interface (asynchronous, 8N1) 3.3V levels (Low=0V, High=3,3V) |
| 3 | USB  Interface |
| 4 | CAN Fieldbus- Interface |
| 5 | 100BaseTX ("Ethernet") |
| 6 | RS422 (differential, full duplex, 4-wire, 3.3V levels) |

### Basic Settings: Type of application layer

| Enum-No. | Meaning |
|---|---|
| 1 | GSV8/6 protocol, as described within this document |
| 2 | CANopen |
| 3 | EtherCAT CoE |
| 4 | ASCII "Monitor" protocol (GSV-6 only) |

### Basic Settings: Flags inside flag value

| Bit-No. | Meaning |
|---|---|
| 0 | =1: Inteface is active and ready to receive |
| 1 | =1: Inteface is active and ready to transmit |
| 2 | =1: Interface has write access |
| 3 | =1: Integer-measuring data value is Binary offset format (valid with App.Enum =1 only)<br>=0: Integer-measuring data value is Signed int format (valid with App.Enum =1 only) |
| 16 | =1: Interface switchable to on- and off-state |
| 17 | =1: Interface used device- or service address (Feldbus) |
| 18 | =1: Address(es) is/are changeable |

This value is communicated in Bits<23:0> of the data value in the Basic Settings

### Extended Settings: Type of data content (Enum)


ME-Meßsysteme GmbH
Neuendorfstr. 18a
16761 Hennigsdorf
Tel.: +49 3302 89824 60
Fax: +49 3302 89824 69
Mail: info@me-systeme.de
Web: www.me-systeme.de

| Enum-No. | Meaning |
|---|---|
| 1 | Mask for setting the Basic-settings flag value: Bit=1: Same BitNo can be set to 1 |
| 2 | Mask for setting the Basic-settings flag value: Bit=1: Same BitNo can be set to 0 |
| 3 | Index-No. of other interface(s), with which it is mutually exclusively present. In every of the 4 bytes an index-No >0 may be written, beginning with the LSbyte. Value =0x00000000 means: Mutually-exclusive with none |
| 4 | Aktive baud rate in Bits/s. Value =0 means: Baudrate not applicable (irrelevant/fixed by physical type) or not changeable |
| 5 | Existing baud rate in bits/s |
| 6 | Number of active service-IDs or (device-) addresses |
| 7 | CAN-ID of command service Host->Device |
| 8 | CAN-ID of command service answers Device-> Host |
| 9 | CAN-ID of measuring value frame Device-> Host |
| 10 | CAN-ID Multicast Host->Devices |
| 11 | CANopen NodeID |
| 16 | Device state, especially with  field bus, see next table |
| 17 | Bits<31:24>: CANopen Transmission-Type. Bits<15:0>: CANopen Event-Timer |
| 18 | Bits<31:16>: CANopen Inhibit-Time. Bits<15:0>: CANopen Heartbeat Timer |

Device state codes (EtherCAT / CANopen)

| Value | Meaning |
|---|---|
| 0 | Interface is switched off |
| 2 | Interface on, State = "Init" / "Stopped" |
| 4 | Interface on, State = "Pre-Operational" |
| 8 | Interface on, State = "Save-Operational" |
| 12 | Interface on, State = "Operational" |

This Code is communicated at type-enum 16 in the extended settings.

### IDs für GSV86readTEDSentry

| ID | Property name and Description |
|---|---|
| 0 | Placeholder in Dictionary, points to first entry index (special value: evaluate NextID only) |
| 1 | TEMPLATE ID |
| 2 | Separator |
| 3 | Select Case—Physical Measurand |

4 Select Case—Full-Scale Electrical Value Precision

5..9 reserviert f. weitere Select-cases u. Sonder-IDs

10 Sensitivity and mapping properties General %Sens Sensitivity of transducer

11 %Sens@Ref Sensitivity of transducer at reference conditions

12 %Reffreq Reference frequency (f ref)

13 %RefTemp Reference temperature (T ref)

14 %Sign Phase inversion (0° or 180°)

15 %Direction Direction, or axis, of sensitivity (x, y, or z)

16 %MapMeth Mapping Method of physical to electrical units

17 %MinPhysVal Minimum value of physical measurement/control range

18 %MaxPhysVal Maximum value of physical measurement/control range

19 %MinElecVal Minimum value of electrical signal range

20 %MaxElecVal Maximum value of electrical signal range

21 Strain/Bridge %GageType Topology and rosette orientation of gage

22 %BridgeType Type of bridge (quarter, half, or full)

23 %GageFactor Sensitivity of strain gage

24 %GageTransSens Transverse sensitivity of strain gage

25 %GageOffset Zero offset of gage circuit after installation

26 %PoissonCoef Poisson Coefficient of strain gage

27 %YoungsMod Youngs Modulus of material to which gage is attached

28 %GageArea Area of gage element

29 RTD and thermistor %RTDCoef_R0 RTD or thermistor resistance at 0 °C

30 %RTDCoef_A Coefficient A of Callendar Van-Dusen equation for RTDs

31 %RTDCoef_B Coefficient B of Callendar Van-Dusen equation for RTDs

32 %RTDCoef_C Coefficient C of Callendar Van-Dusen equation for RTDs

33 %SteinhartA Coefficient A of Steinhart-Hart equation for thermistors

34 %SteinhartB Coefficient B of Steinhart-Hart equation for thermistors

35 %SteinhartC Coefficient C of Steinhart-Hart equation for thermistors

36 %SelfHeating Coefficient of self-heating, intended for thermistors

37 TC %TCType Thermocouple calibration type (J, K, T, etc.)

38 %CJSource Cold-junction compensation method

39 Electrical signal properties General %ElecSigType Type of electrical signal (enumerated)

40 %RespTime Response Time

41 %ACDCCoupling Coupling of electrical signal (AC or DC)

42 %SensorImped Electrical impedance of sensor (of each element in case of bridge)

43 %DiscSigType Discrete signal type

44 %DiscSigAmpl Discrete signal voltage amplitude

45 %PulseMeasType Pulse signal measurement type (frequency, period, count, etc

46 %Gain Gain of preamplifier

47 %Filter Indicates selectable filter

48 %TempCoef Temperature coefficient

49 Sensitivity and mapping properties Mic/Preamp %Prepolarized Prepolarized (yes or no)

50 %RefPol Polarization voltage

51 %Rin Input resistance of amplifier

52 %Rout Output resistance of amplifier

53 %Cin Input capacitance of amplifier

54 %Cmic Microphone capacitance

55 %Cstray Microphone stray capacitance

56 %Rleakage Microphone leakage resistance

57 %MicType Microphone type

58 %MicSize Microphone size

59 %Resp_Type Frequency response type

60 %RefPress Reference pressure

61 %Equi_Vol Equivalent microphone volume

62 %Gate Gate present

63 Excitation and power %ExciteAmplNom Excitation or power-supply level, nominal

64 %ExciteAmplMin Excitation or power-supply level, minimum

65 %ExciteAmplMax Excitation or power-supply level, maximum

66 %ExciteType Type of excitation or power (DC, AC, or bipolar DC)

67 %ExciteCurrentDraw Maximum current required to power/excite transducer

68 %ExciteFreqNom Excitation signal frequency, nominal

69 %ExciteFreqMin Excitation signal frequency, minimum

70 %ExciteFreqMax Excitation signal frequency, maximum

71 %LoopSupplyMin Supply for current loop transducers, minimum

72 %LoopSupplyMax Supply for current loop transducers, maximum

73 Calibration properties Mg. %CalDate The date of the last calibration

74 %CalInitials Calibration initials

75 %CalPeriod Amount of time recommended between calibrations

76 Sensitivity and mapping properties Calibration table and curves %CalTable_Domain Indicates calibration table domain as electrical or physical

%CalPoint_DomainValue Domain calibration value

%CalPoint_RangeValue Range calibration deviation

%CalCurve_Domain Indicates calibration curve domain as electrical or physical

%CalCurve_PieceStart Start of calibration curve segment

%CalCurve_Power Power of domain value

%CalCurve_Coef Coefficient of polynomial

Transfer function %TF_SZ Single zero

%TF_SP Single pole (low-pass filter of first order)

%TF_KZr Complex zero

%TF_KZq Quality factor parameter Qz of a complex zero

%TF_KPr Complex pole at Fpres (F mounted resonance)

%TF_KPq Quality factor Qp for the complex pole (mounted quality factor)

%TF_HP_S Single zero at 0 and a single pole (high-pass filter)

%TF_SL Constant relative slope

%TF_SZm Single zero dependent on previous property

%TF_SPm Single pole dependent on previous property

%PhaseCorrection Phase correction at the reference condition

%TF_Table_Freq Frequency point value for tabular transfer function

%TF_Table_Ampl Amplitude point value for tabular transfer function

Miscellaneous properties Attached transducer %Attached_MfgrID Manufacturer ID of transducer attached to amplifier

%Attached_ModelNum Model number of transducer attached to amplifier

%Attached_VersionLetter Version letter of transducer attached to amplifier

%Attached_VersionNum Version number of transducer attached to amplifier

%Attached_SerialNum Serial number of transducer attached to amplifier

%System_MfgrID Manufacturer ID of system

%System_ModelNum Model number of system

%System_VersionLetter Version letter of system

%System_VersionNum Version number of system

%System_SerialNum Serial number of system

Sensitivity and mapping properties Miscellaneous %Stiffness Stiffness of transducer

%Mass_below Mass below gage

%Weight Weight of transducer

%TestGain Test gain

%Passive Indicates support of passive mode

111 %PollFreq The frequency with which the host shall update the FR

112 %MeasID Measurand ID

113 %Ccable Capacitance of cable

114 %CableLen Length of cable

115 %Appended_TEDS Indicates if an Appended TEDS exists

116 %Appended_TEDS_location Indicates location of the Appended TEDS if it exists

117 %EMBTPL Indicates that the next portion of the TEDS is in Embedded Template format

118 %DefaultFR Defines the default setting of the FR. Cannot coexist with subproperty Default.

119 %XML XML format

120 %MDEF Prefix for manufacturer-defined parameters

121 %TDL_CHKSUM User template validation checksum

122 %user Freeform TEDS format

123 Grouping properties %PhysicalParameterType Describes the parameter type

124 %MemberIndex Indicate the order in which XdcrChannels are grouped within a PublicXdcr