Kvaser Memorator API

Generated by Doxygen 1.7.3

Sun May 22 2016 19:52:20

Contents

Chapter 1

Data Structure Documentation

1.1 kvmLogEventEx Struct Reference

The union of events used by kvmKmeReadEvent().

`#include <kvmlib.h>`

Data Fields

- uint32 type
- union {
  kvmLogMsgEx msg
  kvmLogRtcClockEx rtc
  kvmLogTriggerEx trig
  kvmLogVersionEx ver
  uint8 raw [128]
  } eventUnion

1.1.1 Detailed Description

The union of events used by kvmKmeReadEvent().

1.1.2 Field Documentation

1.1.2.1 union { ... } eventUnion

1.1.2.2 kvmLogMsgEx msg

A CAN message.

1.1.2.3 uint8 raw[128]

Raw data in a array.

1.1.2.4 kvmLogRtcClockEx rtc

An RTC message.

1.1.2.5 kvmLogTriggerEx trig

A trigger message.

1.1.2.6 uint32 type

kvmLOG_TYPE_xxx, Event types in log

1.1.2.7 kvmLogVersionEx ver

A version message.

The documentation for this struct was generated from the following file:

- kvmlib.h

1.2 kvmLogMsgEx Struct Reference

A CAN message.

`#include <kvmlib.h>`

Data Fields

- uint32 id
- int64 timeStamp
- uint32 channel
- uint32 dlc
- uint32 flags
- uint8 data [64]

1.2.1 Detailed Description

A CAN message.

1.2.2 Field Documentation

1.2.2.1 uint32 channel

The device channel on which the message arrived, 0,1,...

1.2.2.2 uint8 data[64]

Message data (64 bytes)

1.2.2.3 uint32 dlc

The length of the message.

1.2.2.4 uint32 flags

Message flags canMSG_xxx.

1.2.2.5 uint32 id

The message identifier.

1.2.2.6 int64 timeStamp

The timestamp in units of 1 nanosecond.

The documentation for this struct was generated from the following file:

- kvmlib.h

1.3 kvmLogRtcClockEx Struct Reference

A RTC clock message.

```
#include <kvmlib.h>
```

Data Fields

- uint32 calendarTime
- int64 timeStamp

1.3.1 Detailed Description

A RTC clock message.

1.3.2 Field Documentation

1.3.2.1 uint32 calendarTime

RTC date, seconds since 1970-01-01T00:00:00+00:00 (UTC)

1.3.2.2 int64 timeStamp

The timestamp in units of 1 nanosecond.

The documentation for this struct was generated from the following file:

- kvmlib.h

1.4 kvmLogTriggerEx Struct Reference

A trigger message.

```
#include <kvmlib.h>
```

Data Fields

- int32 type
- int32 preTrigger
- int32 postTrigger
- uint32 trigMask
- int64 timeStamp

1.4.1 Detailed Description

A trigger message.

1.4.2 Field Documentation

1.4.2.1 int32 postTrigger

Posttrigger time in milliseconds.

1.4.2.2 int32 preTrigger

Pretrigger time in milliseconds.

1.4.2.3 int64 timeStamp

The timestamp in units of 1 nanosecond.

1.4.2.4 uint32 trigMask

Bitmask with all active triggers.

1.4.2.5 int32 type

The type of trigger TRIGVAR_TYPE_xxx.

The documentation for this struct was generated from the following file:

- kvmlib.h

1.5 kvmLogVersionEx Struct Reference

A version message.

`#include <kvmlib.h>`

Data Fields

- uint32 lioMajor
- uint32 lioMinor
- uint32 fwMajor
- uint32 fwMinor
- uint32 fwBuild
- uint32 serialNumber
- uint32 eanHi
- uint32 eanLo

1.5.1 Detailed Description

A version message.

1.5.2 Field Documentation

1.5.2.1 uint32 eanHi

1.5.2.2 uint32 eanLo

1.5.2.3 uint32 fwBuild

1.5.2.4 uint32 fwMajor

1.5.2.5 uint32 fwMinor

1.5.2.6 uint32 lioMajor

1.5.2.7 uint32 lioMinor

1.5.2.8 uint32 serialNumber

The documentation for this struct was generated from the following file:

- kvmlib.h

Chapter 2

File Documentation

2.1 kvmlib.h File Reference

Library for accessing Kvaser Memorator (2nd generation)

```
#include <windows.h>
#include <stdio.h>
#include <pshpack1.h>
#include <poppack.h>
```

Data Structures

- struct kvmLogMsgEx

  A CAN message.

- struct kvmLogRtcClockEx

  A RTC clock message.

- struct kvmLogTriggerEx

  A trigger message.

- struct kvmLogVersionEx

  A version message.

- struct kvmLogEventEx

  The union of events used by kvmKmeReadEvent().

Defines

kvmDEVICE_xxx

Device type, used to connect to a Memorator device.

- #define kvmDEVICE_MHYDRA 0
- #define kvmDEVICE_MHYDRA_EXT 1

kvmLDF_MAJOR_xxx

Logged data format (LDF) version.

- #define kvmLDF_MAJOR_CAN 3
- #define kvmLDF_MAJOR_CAN64 5

kvmFS_xxx

File system used when formatting disk.

- #define kvmFS_FAT16 0
- #define kvmFS_FAT32 1

kvmFILE_xxx

KME file type, a binary file format representing log data.

- #define kvmFILE_KME24 0
- #define kvmFILE_KME25 1
- #define kvmFILE_KME40 2
- #define kvmFILE_KME50 3

kvm_SWINFO_xxx

Different types of version information that can be extracted using kvmDeviceGetSoftwareInfo()

- #define kvm_SWINFO_KVMLIB 1
- #define kvm_SWINFO_DRIVER 2
- #define kvm_SWINFO_FIRMWARE 3
- #define kvm_SWINFO_DRIVER_PRODUCT 4
- #define kvm_SWINFO_CONFIG_VERSION_NEEDED 5
- #define kvm_SWINFO_CPLD_VERSION 6

canMSG_xxx

The following flags can be found in a kvmLogMsgEx message flags field returned from kvmKmeReadEvent(). All flags and/or combinations of them are meaningful for logged message.

- #define canMSG_RTR 0x0001
- #define canMSG_STD 0x0002
- #define canMSG_EXT 0x0004
- #define canMSG_ERROR_FRAME 0x0020
- #define canMSG_TXACK 0x0040
- #define canMSG_TXRQ 0x0080

- #define canMSGERR_OVERRUN 0x0600
- #define canFDMSG_EDL 0x010000
- #define canFDMSG_FDF 0x010000
- #define canFDMSG_BRS 0x020000
- #define canFDMSG_ESI 0x040000

TRIGVAR_TYPE_xxx

The following trigger types can be found in a kvmLogTriggerEx message type field.

- #define TRIGVAR_TYPE_MSG_ID 0
- #define TRIGVAR_TYPE_MSG_DLC 1
- #define TRIGVAR_TYPE_MSG_FLAG 2
- #define TRIGVAR_TYPE_SIGVAL 3
- #define TRIGVAR_TYPE_EXTERNAL 4
- #define TRIGVAR_TYPE_TIMER 5
- #define TRIGVAR_TYPE_DISK_FULL 6
- #define TRIGVAR_TYPE_STARTUP 9

kvmLOG_TYPE_xxx

Event types in log

- #define kvmLOG_TYPE_INVALID 0
- #define kvmLOG_TYPE_CLOCK 1
- #define kvmLOG_TYPE_MSG 2
- #define kvmLOG_TYPE_TRIGGER 3
- #define kvmLOG_TYPE_VERSION 4

Typedefs

- typedef HANDLE kmeFileHandle
- typedef HANDLE kvmHandle
- typedef signed char int8
- typedef unsigned char uint8
- typedef short int16
- typedef unsigned short uint16
- typedef long int int32
- typedef unsigned long int uint32
- typedef __int64 int64

Enumerations

kvmStatus

Generally, a return code greater than or equal to zero means success. A value less than zero means failure.

- enum kvmStatus {
  kvmOK = 0,
  kvmFail = -1,
  kvmERR_PARAM = -3,
  kvmERR_LOGFILEOPEN = -8,
  kvmERR_NOSTARTTIME = -9,
  kvmERR_NOLOGMSG = -10,
  kvmERR_LOGFILEWRITE = -11,
  kvmEOF = -12,
  kvmERR_NO_DISK = -13,
  kvmERR_LOGFILEREAD = -14,
  kvmERR_QUEUE_FULL = -20,
  kvmERR_CRC_ERROR = -21,
  kvmERR_SECTOR_ERASED = -22,
  kvmERR_FILE_ERROR = -23,
  kvmERR_DISK_ERROR = -24,
  kvmERR_DISKFULL_DIR = -25,
  kvmERR_DISKFULL_DATA = -26,
  kvmERR_SEQ_ERROR = -27,
  kvmERR_FILE_SYSTEM_CORRUPT = -28,
  kvmERR_UNSUPPORTED_VERSION = -29,
  kvmERR_NOT_IMPLEMENTED = -30,
  kvmERR_FATAL_ERROR = -31,
  kvmERR_ILLEGAL_REQUEST = -32,
  kvmERR_FILE_NOT_FOUND = -33,
  kvmERR_NOT_FORMATTED = -34,
  kvmERR_WRONG_DISK_TYPE = -35,
  kvmERR_TIMEOUT = -36,
  kvmERR_DEVICE_COMM_ERROR = -37,
  kvmERR_OCCUPIED = -38,
  kvmERR_USER_CANCEL = -39,
  kvmERR_FIRMWARE = -40,
  kvmERR_CONFIG_ERROR = -41,
  kvmERR_WRITE_PROT = -42 }

Functions

- void kvmInitialize (void)
- kvmStatus kvmGetVersion (int *major, int *minor, int *build)
- kvmStatus kvmGetErrorText (kvmStatus error, char *buf, size_t len)
- kvmStatus kvmClose (kvmHandle h)
- kvmHandle kvmDeviceOpen (int32 cardNr, kvmStatus *status, int32 deviceType)

- kvmStatus kvmDeviceMountKmf (kvmHandle h)
- kvmStatus kvmDeviceMountKmfEx (kvmHandle h, int *ldfMajor, int *ldfMinor)
- kvmHandle kvmKmfOpen (const char *filename, kvmStatus *status, int32 deviceType)
- kvmHandle kvmKmfOpenEx (const char *filename, kvmStatus *status, int32 deviceType, int *ldfMajor, int *ldfMinor)
- kvmStatus kvmKmfValidate (kvmHandle h)
- kvmStatus kvmDeviceFormatDisk (kvmHandle h, int fileSystem, uint32 reserveSpace, uint32 dbaseSpace)
- kvmStatus kvmLogFileGetCount (kvmHandle h, uint32 *fileCount)
- kvmStatus kvmLogFileMount (kvmHandle h, uint32 fileIndx, uint32 *eventCount)
- kvmStatus kvmLogFileDismount (kvmHandle h)
- kvmStatus kvmLogFileGetStartTime (kvmHandle h, uint32 *startTime)
- kvmStatus kvmLogFileGetEndTime (kvmHandle h, uint32 *endTime)
- kvmStatus kvmLogFileGetCreatorSerial (kvmHandle h, uint32 *serialNumber)
- kvmStatus kvmLogFileReadEvent (kvmHandle h, kvmLogEventEx *e)
- kvmStatus kvmLogFileDeleteAll (kvmHandle h)
- kvmStatus kvmDeviceDiskStatus (kvmHandle h, int *present)
- kvmStatus kvmKmfGetUsage (kvmHandle h, uint32 *totalSectorCount, uint32 *usedSectorCount)
- kvmStatus kvmDeviceDiskSize (kvmHandle h, uint32 *diskSize)
- kvmStatus kvmDeviceGetSerialNumber (kvmHandle h, unsigned int *serial)
- kvmStatus kvmDeviceGetSoftwareInfo (kvmHandle h, int32 itemCode, unsigned int *major, unsigned int *minor, unsigned int *build, unsigned int *flags)
- kvmStatus kvmDeviceFlashLeds (kvmHandle h)
- kvmStatus kvmDeviceGetRTC (kvmHandle h, unsigned long *t)
- kvmStatus kvmDeviceSetRTC (kvmHandle h, unsigned long t)
- kvmStatus kvmKmfReadConfig (kvmHandle h, void *buf, size_t buflen, size_t *actual_len)
- kvmStatus kvmKmfWriteConfig (kvmHandle h, void *buf, size_t buflen)
- kvmStatus kvmKmfGetDbaseFile (kvmHandle h, char *path, char *filenamebuf, size_t buflen)
- kvmStatus kvmKmfPutDbaseFile (kvmHandle h, char *filename)
- kvmStatus kvmKmfEraseDbaseFile (kvmHandle h)
- kmeFileHandle kvmKmeOpenFile (const char *filename, kvmStatus *status, int32 fileType)
- kmeFileHandle kvmKmeCreateFile (const char *filename, kvmStatus *status, int32 fileType)
- kvmStatus kvmKmeReadEvent (kmeFileHandle h, kvmLogEventEx *e)
- kvmStatus kvmKmeWriteEvent (kmeFileHandle h, kvmLogEventEx *e)
- kvmStatus kvmKmeCountEvents (kmeFileHandle h, uint32 *eventCount)
- kvmStatus kvmKmeCloseFile (kmeFileHandle h)

2.1.1 Detailed Description

Library for accessing Kvaser Memorator (2nd generation) Copyright 2015 by KVASER AB, SWEDEN WWW: http://www.kvaser.com

This software is furnished under a license and may be used and copied only in accordance with the terms of such license.

Description: Library for accessing Kvaser Memorator (2nd generation). This library is used to extract log data, initialize disk, read and write configuration to a device, handle on device databases and more.

2.1.2 Define Documentation

2.1.2.1 #define canFDMSG_BRS 0x020000

Message is sent/received with bit rate switch (CAN FD)

2.1.2.2 #define canFDMSG_EDL 0x010000

Obsolete, use MSGFLAG_FDF instead.

2.1.2.3 #define canFDMSG_ESI 0x040000

Sender of the message is in error passive mode (CAN FD)

2.1.2.4 #define canFDMSG_FDF 0x010000

Message is an FD message (CAN FD)

2.1.2.5 #define canMSG_ERROR_FRAME 0x0020

Message is an error frame.

2.1.2.6 #define canMSG_EXT 0x0004

Message has an extended ID.

2.1.2.7 #define canMSG_RTR 0x0001

Message is a remote request.

2.1.2.8 #define canMSG_STD 0x0002

Message has a standard ID.

2.1.2.9 #define canMSG_TXACK 0x0040

Message is a TX ACK (msg is really sent)

2.1.2.10 #define canMSG_TXRQ 0x0080

Message is a TX REQUEST (msg is transfered to the CAN controller chip)

2.1.2.11 #define canMSGERR_OVERRUN 0x0600

Message overrun condition occurred.

2.1.2.12 #define kvm_SWINFO_CONFIG_VERSION_NEEDED 5

Returns the version of the binary format (param.lif).

2.1.2.13 #define kvm_SWINFO_CPLD_VERSION 6

Obsolete.

2.1.2.14 #define kvm_SWINFO_DRIVER 2

Returns the used driver version information.

2.1.2.15 #define kvm_SWINFO_DRIVER_PRODUCT 4

Obsolete. Returns the product version information.

2.1.2.16 #define kvm_SWINFO_FIRMWARE 3

Returns the device firmware version information.

2.1.2.17 #define kvm_SWINFO_KVMLIB 1

Returns the version of kvmlib.

2.1.2.18 #define kvmDEVICE_MHYDRA 0

Kvaser Memorator (2nd generation)

2.1.2.19 #define kvmDEVICE_MHYDRA_EXT 1

Kvaser Memorator (2nd generation) with extended data capabilities.

2.1.2.20 #define kvmFILE_KME24 0

Deprecated.

2.1.2.21 #define kvmFILE_KME25 1

Deprecated.

2.1.2.22 #define kvmFILE_KME40 2

Kvaser binary format (KME 4.0)

2.1.2.23 #define kvmFILE_KME50 3

Kvaser binary format (KME 5.0)

2.1.2.24 #define kvmFS_FAT16 0

fat16

2.1.2.25 #define kvmFS_FAT32 1

fat32

2.1.2.26 #define kvmLDF_MAJOR_CAN 3

Used in Kvaser Memorator (2nd generation)

2.1.2.27 #define kvmLDF_MAJOR_CAN64 5

Used in Kvaser Memorator (2nd generation) with extended data capabilities.

2.1.2.28 #define kvmLOG_TYPE_CLOCK 1

The type used in kvmLogRtcClockEx.

2.1.2.29 #define kvmLOG_TYPE_INVALID 0

Invalid MEMOLOG type.

2.1.2.30 #define kvmLOG_TYPE_MSG 2

The type used in kvmLogMsgEx.

2.1.2.31 #define kvmLOG_TYPE_TRIGGER 3

The type used in kvmLogTriggerEx.

2.1.2.32 #define kvmLOG_TYPE_VERSION 4

The type used in kvmLogVersionEx.

2.1.2.33 #define TRIGVAR_TYPE_DISK_FULL 6

Disk is full trigger.

2.1.2.34 #define TRIGVAR_TYPE_EXTERNAL 4

External trigger.

2.1.2.35 #define TRIGVAR_TYPE_MSG_DLC 1

Message DLC trigger.

2.1.2.36 #define TRIGVAR_TYPE_MSG_FLAG 2

Message flag trigger.

2.1.2.37 #define TRIGVAR_TYPE_MSG_ID 0

Message ID trigger.

2.1.2.38 #define TRIGVAR_TYPE_SIGVAL 3

Signal value trigger.

2.1.2.39 #define TRIGVAR_TYPE_STARTUP 9

Startup trigger.

2.1.2.40 #define TRIGVAR_TYPE_TIMER 5

Timer trigger.

2.1.3 Typedef Documentation

2.1.3.1 typedef short int16

2.1.3.2 typedef long int int32

2.1.3.3 typedef __int64 int64

2.1.3.4 typedef signed char int8

2.1.3.5 typedef HANDLE kmeFileHandle

A handle to a KME file.

2.1.3.6 typedef HANDLE kvmHandle

A handle to a Memorator or equivalent KMF file.

2.1.3.7 typedef unsigned short uint16

2.1.3.8 typedef unsigned long int uint32

2.1.3.9 typedef unsigned char uint8

2.1.4 Enumeration Type Documentation

2.1.4.1 enum kvmStatus

Enumerator:

kvmOK OK!

kvmFail Generic error.

kvmERR_PARAM Error in supplied parameters.

kvmERR_LOGFILEOPEN Can't find/open log file.

kvmERR_NOSTARTTIME Start time not found.

kvmERR_NOLOGMSG No log message found.

kvmERR_LOGFILEWRITE Error writing log file.

kvmEOF End of file found.

kvmERR_NO_DISK No disk found.

kvmERR_LOGFILEREAD Error while reading log file.

kvmERR_QUEUE_FULL Queue is full.

kvmERR_CRC_ERROR CRC check failed.

kvmERR_SECTOR_ERASED Sector unexpectadly erased.

kvmERR_FILE_ERROR File I/O error.

kvmERR_DISK_ERROR General disk error.

kvmERR_DISKFULL_DIR Disk full (directory).

kvmERR_DISKFULL_DATA Disk full (data).

kvmERR_SEQ_ERROR Unexpected sequence.

kvmERR_FILE_SYSTEM_CORRUPT File system corrupt.

kvmERR_UNSUPPORTED_VERSION Unsupported version.

kvmERR_NOT_IMPLEMENTED Not implemented.

kvmERR_FATAL_ERROR Fatal error.

kvmERR_ILLEGAL_REQUEST Illegal request.

kvmERR_FILE_NOT_FOUND File not found.

kvmERR_NOT_FORMATTED Disk not formatted.

kvmERR_WRONG_DISK_TYPE Wrong disk type.

kvmERR_TIMEOUT Timeout.

kvmERR_DEVICE_COMM_ERROR Device communication error.

kvmERR_OCCUPIED Device occupied.

kvmERR_USER_CANCEL User abort.

kvmERR_FIRMWARE Firmware error.

kvmERR_CONFIG_ERROR Configuration error.

kvmERR_WRITE_PROT Disk is write protected.

2.1.5 Function Documentation

2.1.5.1 kvmStatus kvmClose (kvmHandle *h*)

C#

static Kvmlib.Status Close(Handle^ h);

Close the connection to the Memorator (device or file) opened with kvmDeviceOpen() or kvmDeviceMountKmf(). The handle becomes invalid.

Parameters

in	*h*	An open kvmHandle.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

See also

kvmDeviceMountKmf(), kvmDeviceOpen(), kvmDeviceMountKmfEx()

2.1.5.2 kvmStatus kvmDeviceDiskSize (kvmHandle *h*, uint32 * *diskSize*)

C#

static Kvmlib.Status DeviceDiskSize(Handle^ h, out Int32 diskSize);

Get disk size, reported in number of (512 byte) sectors.

Parameters

in	*h*	An open kvmHandle.
out	*diskSize*	Disk size in number of (512 byte) sectors.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.3 kvmStatus kvmDeviceDiskStatus (kvmHandle *h*, int * *present*)

C#

static Kvmlib.Status DeviceDiskStatus(Handle^ h, out Int32 present);

Check if the SD memory card is present.

Note

This function is not supported by all devices.

Parameters

in	*h*	An open kvmHandle.
out	*present*	Non-zero means that SD memory card is present.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.4 kvmStatus kvmDeviceFlashLeds (kvmHandle *h*)

C#

static Kvmlib.Status DeviceFlashLeds(Handle^ h);

Flash all LEDs on the opened Memorator device

Parameters

in	*h*	An open kvmHandle.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.5 kvmStatus kvmDeviceFormatDisk (kvmHandle *h*, int *fileSystem*, uint32 *reserveSpace*, uint32 *dbaseSpace*)

C#

static Kvmlib.Status DeviceFormatDisk(Handle^ h, Int32 filesystem, Int32 reserveSpace, Int32 dbaseSpace);

Format the SD memory card in a connected Memorator.

An item code specifying the type of version to get. kvm_SWINFO_xxx

Parameters

in	*h*	An open kvmHandle.
in	*fileSystem*	A filesystem code, kvmFS_xxx, specifying the type of filesystem to format to.
in	*reserveSpace*	Space to reserve for user files, in MB.
in	*dbaseSpace*	Space to reserve for database files, in MB.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

See also

kvmDeviceDiskSize

2.1.5.6 kvmStatus kvmDeviceGetRTC (kvmHandle *h*, unsigned long * *t*)

C#

static Kvmlib.Status DeviceGetRTC(Handle^ h, out Int32 t);

Get date and time from the RTC chip. The time is returned in standard unix time (number of seconds since 1970-01-01T00:00:00+00:00). Only for device handles.

Parameters

in	*h*	An open kvmHandle.
out	*t*	Time in Unix time.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.7 kvmStatus kvmDeviceGetSerialNumber (kvmHandle *h*, unsigned int * *serial*)

C#

static Kvmlib.Status DeviceGetSerialNumber(Handle^ h, out Int32 serial);

Get serial number related to the Memorator handle.

Parameters

in	*h*	An open kvmHandle.
out	*serial*	Serial number of connected device.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.8 kvmStatus kvmDeviceGetSoftwareInfo (kvmHandle *h*, int32 *itemCode*, unsigned int * *major*, unsigned int * *minor*, unsigned int * *build*, unsigned int * *flags*)

C#

static Kvmlib.Status DeviceGetSoftwareInfo(Handle^ h, VersionInfo itemCode, out Int32 major, out Int32 minor, out Int32 build, out Int32 flags);

Get software version information.

Parameters

in	*h*	An open kvmHandle.
in	*itemCode*	An item code specifying the type of version to get. kvm_-SWINFO_xxx
out	*major*	Major version number
out	*minor*	Minor version number
out	*build*	Build number
out	*flags*	For internal use only

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.9 kvmStatus kvmDeviceMountKmf (kvmHandle *h*)

C#

static Kvmlib.Status DeviceMountKmf(Handle^ h);

Mount the log area on the SD card on a connected Kvaser Memorator.

Note

Must be called after kvmDeviceOpen before any subsequent log operations are called.

param[in] h An open kvmHandle.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

kvmClose(), kvmDeviceOpen()

2.1.5.10 kvmStatus kvmDeviceMountKmfEx (kvmHandle *h,* int * *ldfMajor,* int * *ldfMinor*)

C#

static Kvmlib.Status DeviceMountKmfEx(Handle^ h, out Int32 ldfMajor, out Int32 ldfMinor);

Mount the log area on the SD card on a connected Kvaser Memorator and return the logger data format (LDF) version.

Note

Must be called after kvmDeviceOpen before any subsequent log operations are called.

param[in] h An open kvmHandle. param[out] ldfMajor Major LDF version kvmLDF_-MAJOR_xxx. param[out] ldfMinor Minor LDF version.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

kvmClose(), kvmDeviceOpen()

2.1.5.11 kvmHandle kvmDeviceOpen (int32 *cardNr*, kvmStatus * *status*, int32 *deviceType*)

C#

static Kvmlib.Handle DeviceOpen(Int32 memoNr, out Status status, DeviceType deviceType);

Connect to a Memorator device and obtain a handle for subsequent device operations. The argument *cardNr* is the Card Number property (decreased by one) displayed in Kvaser Hardware.

Parameters

in	*cardNr*	Card number
out	*status*	kvmOK if completely successful, kvmERR_xxx (negative) if failure
in	*deviceType*	kvmDEVICE_xxx

Returns

Returns an open handle to a Memorator on success.

See also

kvmClose(), kvmLogFileMount()

2.1.5.12 kvmStatus kvmDeviceSetRTC (kvmHandle *h*, unsigned long *t*)

C#

static Kvmlib.Status DeviceSetRTC(Handle^ h, Int32 t);

Set date and time in the RTC. The time is returned in standard Unix time (number of seconds since 1970-01-01T00:00:00+00:00). Only for device handles.

Parameters

in	*h*	An open kvmHandle.
in	*t*	Time in Unix time.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.13 kvmStatus kvmGetErrorText (kvmStatus *error*, char * *buf*, size_t *len*)

C#

static Kvmlib.Status GetErrorText(Status error, out String^ buf);

Convert a kvmStatus errorcode to a text.

Parameters

in	*error*	The error code to convert.
out	*buf*	Buffer to receive error text.
in	*len*	Buffer size in bytes.

2.1.5.14 kvmStatus kvmGetVersion (int * *major*, int * *minor*, int * *build*)

This function returns the version of the KVMLIB API DLL (kvmlib.dll).

Parameters

out	*major*	Major version number.
out	*minor*	Minor version number.
out	*build*	Build number.

Returns

version number of kvmlib.dll

2.1.5.15 void kvmInitialize (void)

C#

static void Initialize(void);

This function must be called before any other functions are used. It will initialize the memolib library.

2.1.5.16 kvmStatus kvmKmeCloseFile (kmeFileHandle *h*)

C#

static Kvmlib.Status KmeCloseFile(Handle^ h);

Close an open KME file opened with kvmKmeOpenFile() or created with kvmKmeCreateFile(). The handle becomes invalid.

Parameters

in	*h*	An open handle to a KME file.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

See also

kvmKmeOpenFile(), kvmKmeCreateFile()

2.1.5.17 kvmStatus kvmKmeCountEvents (kmeFileHandle *h*, uint32 * *eventCount*)

C#

static Kvmlib.Status KmeCountEvents(Handle^ h, out Int32 eventCount);

Count the number of events in a KME file.

Parameters

in	*h*	An open handle to a KME file.
out	*eventCount*	Approximate number of events in a KME file.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

See also

kvmKmeOpenFile(), kvmKmeCreateFile()

2.1.5.18 kmeFileHandle kvmKmeCreateFile (const char * *filename*, kvmStatus * *status*, int32 *fileType*)

C#

static Kvmlib.Handle KmeCreateFile(String^ filename, out Status status, Int32 filetype);

Open a KME file for writing and obtain a handle for subsequent operations. Note that kvmKmeCreateFile() will overwrite any existing file and that kvmFILE_KME24 and kvmFILE_KME25 are deprecated formats. Please use kvmFILE_KME40.

Parameters

in	*filename*	The full path and name of the KME file, e.g. C:\temp\myfile.kme
in	*fileType*	kvmFILE_xxx
out	*status*	kvmOK (zero) if success kvmERR_xxx (negative) if failure

Returns

Returns an open handle to a KME file on success.

See also

kvmKmeWriteEvent(), kvmKmeCountEvents(), kvmKmeCloseFile()

2.1.5.19 kmeFileHandle kvmKmeOpenFile (const char * *filename*, kvmStatus * *status*, int32 *fileType*)

C#

static Kvmlib.Handle KmeOpenFile(String^ filename, out Status status, Int32 filetype);

Open a KME file for reading and obtain a handle for subsequent operations.

Parameters

in	*filename*	The full path and name of the KME file, e.g. C:\temp\myfile.kme
in	*fileType*	kvmFILE_xxx
out	*status*	kvmOK (zero) if success kvmERR_xxx (negative) if failure

Returns

Returns an open handle to a KME file on success.

See also

kvmKmeReadEvent(), kvmKmeCountEvents(), kvmKmeCloseFile()

2.1.5.20 kvmStatus kvmKmeReadEvent (kmeFileHandle *h*, kvmLogEventEx * *e*)

C#

static Kvmlib.Status KmeReadEvent(Handle^ h, out Log^ e);

Read an event from a KME file opened with kvmKmeOpenFile().

Parameters

in	*h*	An open handle to a KME file.
out	*e*	Event from a KME file.

Returns

kvmOK (zero) if success
kvmERR_NOLOGMSG on EOF
kvmERR_xxx (negative) if failure

See also

kvmKmeOpenFile(), kvmKmeCountEvents(), kvmKmeCloseFile()

2.1.5.21 kvmStatus kvmKmeWriteEvent (kmeFileHandle *h*, kvmLogEventEx * *e*)

C#

static Kvmlib.Status KmeWriteEvent(Handle^ h, Log^ e);

Write an event to a KME file created with kvmKmeCreateFile().

Parameters

in	*h*	An open handle to a KME file.
in	*e*	Event to write.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

See also

kvmKmeCreateFile(), kvmKmeCountEvents(), kvmKmeCloseFile()

2.1.5.22 kvmStatus kvmKmfEraseDbaseFile (kvmHandle *h*)

C#

static Kvmlib.Status KmfEraseDbaseFile(Handle^ h, Int32 filenumber);

Erase the database file.

Parameters

in	*h*	An open kvmHandle.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.23 kvmStatus kvmKmfGetDbaseFile (kvmHandle *h*, char * *path*, char * *filenamebuf*, size_t *buflen*)

C#

static Kvmlib.Status KmfGetDbaseFile(Handle^ h, String^ path, out String^ filenamebuf);

Read the database file. The database will be extracted to path and the name of the created file copied to filenamebuf.

Parameters

in	*h*	An open kvmHandle.
in	*path*	The path where the database file will be stored.
out	*filenamebuf*	The filename of the database. (should be greater then 12 bytes)
in	*buflen*	The lenght of filenamebuf

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

See also

kvmKmfPutDbaseFile()

2.1.5.24 kvmStatus kvmKmfGetUsage (kvmHandle *h*, uint32 * *totalSectorCount*, uint32 * *usedSectorCount*)

C#

static Kvmlib.Status KmfGetUsage(Handle^ h, out Int32 totalSectorCount, out Int32 usedSectorCount);

Get disk usage statistics, reported in number of (512 byte) sectors.

Parameters

in	*h*	An open kvmHandle.
out	*totalSector-Count*	Total number of sectors devoted for logging
out	*usedSector-Count*	Number of logging sectors used

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.25 kvmHandle kvmKmfOpen (const char * *filename*, kvmStatus * *status*, int32 *deviceType*)

C#

static Kvmlib.Handle KmfOpen(String^ filename, out Status status, DeviceType deviceType);

Open a KMF file on a hard disk or SD card reader and obtain a handle for subsequent operations. *deviceType* is the device type that generated the file.

Parameters

in	*filename*	KMF filename
out	*status*	kvmOK if successful, otherwise kvmERR_xxx
in	*deviceType*	kvmDEVICE_xxx

Returns

Returns an open handle to a Memorator on success.

See also

kvmClose(), kvmDeviceOpen()

2.1.5.26 kvmHandle kvmKmfOpenEx (const char * *filename*, kvmStatus * *status*, int32 *deviceType*, int * *ldfMajor*, int * *ldfMinor*)

C#

static Kvmlib.Handle KmfOpenEx(String^ filename, out Status status, DeviceType deviceType, out Int32 ldfMajor, out Int32 ldfMinor);

Open a KMF file on a hard disk or SD card reader and obtain a handle for subsequent operations and return the logger data format (LDF) version. *deviceType* is the device type that generated the file.

Parameters

in	*filename*	KMF filename
out	*status*	kvmOK if successful, otherwise kvmERR_xxx
in	*deviceType*	kvmDEVICE_xxx
out	*ldfMajor*	Major LDF version kvmLDF_MAJOR_xxx.
out	*ldfMinor*	Minor LDF version.

Returns

Returns an open handle to a Memorator on success.

See also

kvmClose(), kvmDeviceOpen()

2.1.5.27 kvmStatus kvmKmfPutDbaseFile (kvmHandle *h*, char * *filename*)

C#

static Kvmlib.Status KmfPutDbaseFile(Handle^ h, String^ filename);

Write the database file

Parameters

in	*h*	An open kvmHandle.
in	*filename*	The full path and name of the file, e.g. C:\temp\myfile.data Note that the filename will be trucated to an 8.3 filename.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

See also

kvmKmfGetDbaseFile()

2.1.5.28 kvmStatus kvmKmfReadConfig (kvmHandle *h*, void * *buf*, size_t *buflen*, size_t * *actual_len*)

C#

static Kvmlib.Status KmfReadConfig(Handle^ h, out array<Byte>^ buf);

Read binary configuration data (param.lif) from a KMF file.

Parameters

in	*h*	An open kvmHandle.
out	*buf*	A pointer to buffer where the configuration (param.lif) will be written.
in	*buflen*	The length of the buffer buf.
out	*actual_len*	The actual length of the configuration written to buf.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.29 kvmStatus kvmKmfValidate (kvmHandle *h*)

C#

static Kvmlib.Status KmfValidate(Handle^ h);

Check for errors

Parameters

in	*h*	An open kvmHandle.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.30 kvmStatus kvmKmfWriteConfig (kvmHandle *h*, void * *buf*, size_t *buflen*)

C#

static Kvmlib.Status KmfWriteConfig(Handle^ h, array<Byte>^ buf);

Write binary configuration data (param.lif) to a KMF file.

Parameters

in	*h*	An open kvmHandle.
in	*buf*	A pointer to buffer containing the configuration (param.lif) to be written.
in	*buflen*	The length of the buffer buf.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.31 kvmStatus kvmLogFileDeleteAll (kvmHandle *h*)

C#

static Kvmlib.Status LogFileDeleteAll(Handle^ h);

Delete all log files from a Memorator.

Parameters

in	*h*	An open kvmHandle.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.32 kvmStatus kvmLogFileDismount (kvmHandle *h*)

C#

static void LogFileDismount(Handle^ h);

Dismount the log file opened with kvmLogFileMount(). The handle will stay valid.

Parameters

in	*h*	An open kvmHandle.

2.1.5.33 kvmStatus kvmLogFileGetCount (kvmHandle *h*, uint32 * *fileCount*)

C#

static Kvmlib.Status LogFileGetCount(Handle^ h, out Int32 fileCount);

Count the number of log files

Parameters

in	*h*	An open kvmHandle.
out	*fileCount*	The number of log files on disk.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.34 kvmStatus kvmLogFileGetCreatorSerial (kvmHandle *h*, uint32 * *serialNumber*)

C#

static Kvmlib.Status LogFileGetCreatorSerial(Handle^ h, out Int32 serialNumber);

Get the serialnumber of the interface that created the log file.

Parameters

in	*h*	An open kvmHandle.
out	*serialNumber*	The serialnumber of the interface that created the log file.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.35 kvmStatus kvmLogFileGetEndTime (kvmHandle *h*, uint32 * *endTime*)

C#

static Kvmlib.Status LogFileGetEndTime(Handle^ h, out Int32 endTime);

Get the time of the first event in the log file. The time is returned in standard unix time format (number of seconds since 1970-01-01T00:00:00+00:00).

Parameters

in	*h*	An open kvmHandle.
out	*endTime*	The time of the last event in the log file (UTC)

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.36 kvmStatus kvmLogFileGetStartTime (kvmHandle *h*, uint32 * *startTime*)

C#

static Kvmlib.Status LogFileGetStartTime(Handle^ h, out Int32 startTime);

Get the time of the first event in the log file. The time is returned in standard unix time format (number of seconds since 1970-01-01T00:00:00+00:00).

Parameters

in	*h*	An open kvmHandle.
out	*startTime*	The time of the first event in the log file (UTC)

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

2.1.5.37 kvmStatus kvmLogFileMount (kvmHandle *h*, uint32 *fileIndx*, uint32 * *eventCount*)

C#

static Kvmlib.Status LogFileMount(Handle^ h, Int32 fileIndx, out Int32 eventCount);

Mount the log file with the specified index. The index starts at 0. The approximate number of events in the log file is returned.

Parameters

in	*h*	An open kvmHandle.
in	*fileIndx*	Index of the log file to open.
out	*eventCount*	The approximate number of events in the log file

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure

See also

kvmLogFileDismount()

2.1.5.38 kvmStatus kvmLogFileReadEvent (kvmHandle *h*, kvmLogEventEx * *e*)

C#

static Kvmlib.Status LogFileReadEvent(Handle^ h, out Log^ e);

Read an event from a log file opened with kvmLogFileMount(). The next call to kvmLogFileReadEvent() will read the next event.

Parameters

in	*h*	An open kvmHandle.
out	*e*	Event from log file.

Returns

kvmOK (zero) if success
kvmERR_xxx (negative) if failure