Software specification for *Kontron EAPI*

Version 3.0

July 1, 2014

CONTENTS

1. REVISION HISTORY 6

2. INTRODUCTION 7

2.1 Purpose of this document 7

2.2 General Information 7

2.2.1 Overview 7

2.2.2 Architecture 7

2.2.3 Supported platforms 7

3. API 9

3.1 General assumptions 9

3.1.1 Return codes 9

3.1.2 Parameters and memory allocation 9

3.1.3 String data 9

3.2 Initialization 10

3.2.1 KEApiLibInitialize 10

3.2.2 KEApiLibUnInitialize 10

3.2.3 KEApiGetLibVersion 10

3.3 General information 12

3.3.1 KEApiGetBoardInfo 12

3.3.2 KEApiGetBootCounter 12

3.3.3 KEApiSystemUpTime 13

3.3.4 KEApiGetIntruderStatus 13

3.3.5 KEApiResetIntruderStatus 14

3.3.6 KEApiGetPBITResult 14

3.3.7 KEApiClearPBITResult 15

3.4 CPU 16

3.4.1 KEApiGetCpuFreq 16

3.4.2 KEApiGetCpuInfo 16

3.4.3 KEApiGetCpuPerformance 17

3.5 Memory 18

3.5.1 KEApiGetMemoryInfo 18

3.6 Hard disks and mount points 19

3.6.1 KEApiGetDiskDriveCount 19
3.6.2 KEApiGetDiskDriveList 19
3.6.3 KEApiGetDiskDriveSMARTAttrCount 20
3.6.4 KEApiGetDiskDriveSMARTAttrs 20
3.6.5 KEApiGetMountPointCount 21
3.6.6 KEApiGetMountPointList 21

3.7 Battery 23
3.7.1 KEApiGetBatteryCount 23
3.7.2 KEApiGetBatteryInfo 23
3.7.3 KEApiGetBatteryState 24

3.8 Performance 26
3.8.1 KEApiPerformanceStateCaps 26
3.8.2 KEApiGetPerformanceStateDescription 26
3.8.3 KEApiGetPerformanceState 27
3.8.4 KEApiSetPerformanceState 27

3.9 Temperature sensors 29
3.9.1 KEApiGetTempSensorCount 29
3.9.2 KEApiGetTempSensorValue 29
3.9.3 KEApiGetTempSensorValueList 30
3.9.4 KEApiGetTempSensorInfo 30

3.10 Voltage sensors 32
3.10.1 KEApiGetVoltageSensorCount 32
3.10.2 KEApiGetVoltageSensorValue 32
3.10.3 KEApiGetVoltageSensorValueList 32
3.10.4 KEApiGetVoltageSensorInfo 33

3.11 Fan Sensors 35
3.11.1 KEApiGetFanSensorCount 35
3.11.2 KEApiGetFanSensorValue 35
3.11.3 KEApiGetFanSensorValueList 35
3.11.4 KEApiGetFanSensorInfo 36

3.12 Display 38
3.12.1 KEApiGetDisplayCount 38
3.12.2 KEApiGetBacklightValue 38
3.12.3 KEApiSetBacklightValue 38

3.13 Network and PCI devices 40
3.13.1 KEApiGetNetworkDeviceCount 40

3.13.2 KEApiGetNetworkDeviceList 40
3.13.3 KEApiGetPciDeviceCount 41
3.13.4 KEApiGetPciDeviceList 41

3.14 Storage area 43
3.14.1 KEApiGetStorageCount 43
3.14.2 KEApiGetStorageSize 43
3.14.3 KEApiStorageRead 43
3.14.4 KEApiStorageWrite 44

3.15 I2C 46
3.15.1 KEApiGetI2cBusCount 46
3.15.2 KEApiI2cXfer 46
3.15.3 KEApiI2cProbe 47

3.16 SPI 49
3.16.1 KEApiGetSpiBusCount 49
3.16.2 KEApiSpiXfer 49

3.17 SMBus 51
3.17.1 KEApiGetSmbusCount 51
3.17.2 KEApiSmbusQuickCommand 51
3.17.3 KEApiSmbusSendByte 52
3.17.4 KEApiSmbusReceiveByte 52
3.17.5 KEApiSmbusWriteByte 53
3.17.6 KEApiSmbusReadByte 54
3.17.7 KEApiSmbusWriteWord 54
3.17.8 KEApiSmbusReadWord 55
3.17.9 KEApiSmbusWriteBlock 56
3.17.10 KEApiSmbusReadBlock 56

3.18 GPIO 58
3.18.1 KEApiGetGpioPortCount 58
3.18.2 KEApiGetGpioPortDirectionCaps 58
3.18.3 KEApiGetGpioPortDirections 59
3.18.4 KEApiSetGpioPortDirections 59
3.18.5 KEApiGetGpioPortLevels 60
3.18.6 KEApiSetGpioPortLevels 60

3.19 Watchdog 62
3.19.1 KEApiWatchdogGetCaps 62
3.19.2 KEApiWatchdogSetup 63

3.19.3 KEApiWatchdogEnable 63
3.19.4 KEApiWatchdogTrigger 64
3.19.5 KEApiWatchdogDisable 64
3.19.6 KEApiWatchdogGetExpired 64
3.19.7 KEApiWatchdogClearExpired 65
3.19.8 KEApiWatchdogWaitUntilExpired 65

4. APPENDIX A: SPECIFICATION CHANGES 66

4.1 Changes from KEAPI release 2.0: 66

5. ABOUT KONTRON 67

1. Revision history

Author	Date	Change summary	Version
Evgeny Denisov	*01 July 2014*	Initial Release	*3.0*

Legal notice:

2. Introduction

2.1 Purpose of this document

This document describes Kontron Embedded API – a software library that enables programmers to easily create their applications for monitoring and control hardware resources of Kontron boards, modules, systems and platforms.

2.2 General Information

2.2.1 Overview

KEAPI is distributed as a static and/or dynamic-link (or shared) library so it can be used by any application developed in C, C++ or higher programming languages simply by linking to the project. The KEAPI library is delivered with Board Support Package (BSP) for any Kontron platform and provides unified interface to hardware drivers and OS-independent API to get platform information.

KEAPI library provides a set of functions for

- Obtaining basic information about the system
- Performance control
- Peripheral devices monitoring (hard disks, network, PCI devices)
- Sensors (temperature, voltage, fans) monitoring
- Power monitoring (batteries)
- Display (backlight) control
- Access to EEPROM user storage area(s)
- Serial bus (I2C, SMBus, SPI) communication
- Hardware Watchdog Timer

KEAPI is compliant and includes wrapper implementation for PICMG EAPI (http://www.picmg.org/pdf/COM_EAPI_R1_0.pdf) and JIDA32 (v1.9a or above).

2.2.2 Architecture

Kontron EAPI is a layer between OS/hardware drivers and user application. KEAPI is a pure library so multiple applications can use it simultaneously. Most of API calls are thread-safe except those API to device drivers which require single-thread access (explicitly noted in specific platform documentation).

The KEAPI is delivered as a C/C++ library set with corresponding C-header files:

- KEAPI is defined in `keapi.h`
- PICMG EAPI is defined in `EApi.h`
- Jida32 interfaces are defined in `Jida.h`

Applications written in High Level Languages (C#, Java) can use bindings (Native Methods, JNI) to the library

Some platforms (such as Linux) may require special privileges to access device drivers. See OS application notes for details.

2.2.3 Supported platforms

KEAPI interface is available for various KONTRON systems running different operating systems. Supported hardware architectures are:

- **Intel X86 32bit and 64bit**
- **AMD X86 32bit and 64bit**
- **ARM**

Supported target operating systems:

- **Linux (Kontron Linux)**

- **Microsoft Windows 7/8, WES7/8**
- **Microsoft Windows Embedded Compact 7, 2013**
- **WindRiver VxWorks 6.x**
- **Android 4.x**
- **QNX**

3. API

Before using any of KEAPI functions, KEAPI has to be initialized and connection to the board has to be established by calling the **KEApiLibInitialize()** function. When KEAPI is no longer needed, the **KEApiLibUnInitialize()** function should be called.

3.1 General assumptions

If everything goes well, all KEAPI functions return `KEAPI_RET_SUCCESS`. If some error occurs, the error code is returned.

3.1.1 Return codes

`KEAPI_RET_ERROR`	Unknown or internal error.
`KEAPI_RET_PARAM_ERROR`	Wrong parameter value
`KEAPI_RET_PARAM_NULL`	Parameter is NULL where it is not allowed
`KEAPI_RET_BUFFER_OVERFLOW`	Buffer overflow (probably configuration error)
`KEAPI_RET_SETTING_ERROR`	Error while setting value or feature (enable, disable)
`KEAPI_RET_RETRIEVAL_ERROR`	Error while retrieving information
`KEAPI_RET_WRITE_ERROR`	Cannot write
`KEAPI_RET_READ_ERROR`	Cannot read
`KEAPI_RET_MALLOC_ERROR`	Memory allocation failed
`KEAPI_RET_LIBRARY_ERROR`	Exported function could not be loaded from library
`KEAPI_RET_WMI_ERROR`	Problems while reading from WMI
`KEAPI_RET_NOT_INITIALIZED`	KEAPI library is not initialized
`KEAPI_RET_PARTIAL_SUCCESS`	Part of requested information couldn't be retrieved. Returned information isn't complete(buffer is not enough).
`KEAPI_RET_FUNCTION_NOT_SUPPORTED`	Function is not supported on current platform/HW
`KEAPI_RET_FUNCTION_NOT_IMPLEMENTED`	Function is not yet implemented
`KEAPI_RET_BUSY_COLLISION`	Bus/Device Busy or Arbitration Error/Collision Error
`KEAPI_RET_BUS_ERROR`	No acknowledge on address or bus error during operation
`KEAPI_RET_HW_TIMEOUT`	Timeout occurred while accessing to device
`KEAPI_RET_CANCELLED`	Operation is cancelled
`KEAPI_RET_PERMISSION_DENIED`	Insufficient user permissions (cannot access the device)

3.1.2 Parameters and memory allocation

Parameters where a value is returned (outputs) are defined as pointers.

Memory for structures and variables that have to be filled by KEAPI functions **must be pre-allocated** by the client application. KEAPI by itself **doesn't allocate memory**.

3.1.3 String data

All fields or string buffers in KEAPI data structures are represented as fixed-size (`KEAPI_MAX_STR`) arrays of `char`. Any string shall be zero-terminated C-string. Thus maximal string length for KEAPI data is `KEAPI_MAX_STR-1`

3.2 Initialization

3.2.1 KEApiLibInitialize

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiLibInitialize();
```

Description:

Initialization of Kontron EAPI.

Parameters: None

Returns:

`KEAPI_RET_SUCCESS` – successfully initialized

`KEAPI_RET_ERROR` – other error

3.2.2 KEApiLibUnInitialize

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiLibUnInitialize();
```

Description:

Kontron EAPI uninitialization.

Parameters: None

Returns:

`KEAPI_RET_SUCCESS`
`KEAPI_RET_NOT_INITIALIZED`

`KEAPI_RET_ERROR` – other error

3.2.3 KEApiGetLibVersion

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetLibVersion (
     PKEAPI_VERSION_DATA pVersion
);
```

Description:

Get KEAPI revision.

Parameters:

in/out	Parameter name	Description
out	*pVersion*	Buffer to receive KEAPI revision information

Structure used:

```
typedef struct Keapi_Version_Data {
     char        versionText[KEAPI_MAX_STR]; // information string
     uint32_t    version;        // 4-byte code (major, minor, patch, build)
} KEAPI_VERSION_DATA, *PKEAPI_VERSION_DATA;
```

Returns:

`KEAPI_RET_SUCCESS`

3.3 General information

3.3.1 KEApiGetBoardInfo

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetBoardInfo (
     PKEAPI_BOARD_INFO pBoardInfo
);
```

Description:

Provides information about Kontron motherboard.

Parameters:

in/out	Parameter name	Description
out	*pBoardInfo*	Returned board info structure **KEAPI_BOARD_INFO**

Structure used:

NOTE: Some fields may be not available (not applicable). String fields which are N/A should be zero length strings (field[0] == '\0'). Integer fields shall be (?int?_t)(-1) then.

```
typedef struct Keapi_Board_Info
{
     char         boardManufacturer[KEAPI_MAX_STR]; // Board manufacturer
     char         boardName[KEAPI_MAX_STR];    // Board name
     char         boardSerialNumber[KEAPI_MAX_STR]; // Board serial number
     char         hardwareVersion[KEAPI_MAX_STR]; // hardware revision
                                                   in text form */
     int64_t      manufacturingDate;           // Board Manufacturing date as
                                                  POSIX timestamp (time_t)
     int64_t      lastRepairDate;              // Date that the system was
                                                 last repaired or
                                                 refurbished.
                                                 Valid only if later
                                                 than the manufacturing date.
                                                 POSIX timestamp (time_t)
     char         carrierInfo[KEAPI_MAX_STR];  // Carrier name and version
     char         firmwareVersion[KEAPI_MAX_STR]; // Bootloader/BIOS version
     int64_t      firmwareDate;                // Bootloader/BIOS date as
                                                  POSIX timestamp (time_t)
} KEAPI_BOARD_INFO, *PKEAPI_BOARD_INFO;
```

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.3.2 KEApiGetBootCounter

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetBootCounter (
      int32_t *pBootCount
);
```

Description:

Provides information about number of boot cycles within the board’s lifetime.

Parameters:

in/out	Parameter name	Description
Out	*pBootCount*	Number of boot cycles

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.3.3 KEApiSystemUpTime

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSystemUpTime (
      int32_t *pSystemUpTime
);
```

Description:

Provides time left since last boot in seconds.

Parameters:

in/out	Parameter name	Description
out	*pSystemUpTime*	Pointer to a variable that receives system running time in seconds

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.3.4 KEApiGetIntruderStatus

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetIntruderStatus (
      int32_t *pIntruderStatus
);
```

Description:

Provides actual information whether computer case was opened or not.

Parameters:

in/out	Parameter name	Description

out	*pIntruderStatus*	Pointer to a variable that receives actual intruder status, possible status values: KEAPI_INTRUDER_STATUS_CASE_CLOSED, 0 KEAPI_INTRUDER_STATUS_CASE_OPENED , 1

Returns:

```
KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error
```

3.3.5 KEApiResetIntruderStatus

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiResetIntruderStatus (void);
```

Description:

Resets the case intruder status.

Parameters:

None

Returns:

```
KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error
```

3.3.6 KEApiGetPBITResult

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetPBITResult (
      uint32_t *pResult
      uint32_t *pCumulativeResult
);
```

Description:

Get result of Power-on built-in test (PBIT). Valid for platforms with PBIT support.

Parameters:

in/out	Parameter name	Description
out	*pResult*	Latest status
out	*pCumulativeResult*	Cumulative status

Returns:

```
KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error
```

3.3.7 KEApiClearPBITResult

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiClearPBITResult (void);
```

Description:

Resets latest result of Power-on built-in test (PBIT). Valid for platforms with PBIT support.

Parameters:

None

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.4 CPU

3.4.1 KEApiGetCpuFreq

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetCpuFreq (
      int32_t coreNr,
      int8_t freqType,
      int32_t *pFrequency
);
```

Description:

Provides information about CPU core frequency

minimal supported CPU core frequency: KEAPI_CPU_FREQUENCY_MIN

maximal supported CPU core frequency: KEAPI_CPU_FREQUENCY_MAX

current CPU core frequency: KEAPI_CPU_FREQUENCY_CURRENT

Turbo frequency (e.g. Intel (C) Turbo Boost Technology): KEAPI_CPU_FREQUENCY_TURBO

Parameters:

in/out	Parameter name	Description
in	*coreNr*	CPU core number (zero based, global counter through all CPUs).
In	*freqType*	Type of frequency (minimal (KEAPI_CPU_FREQUENCY_MIN), maximal (KEAPI_CPU_FREQUENCY_MAX), current (KEAPI_CPU_FREQUENCY_CURRENT), turbo (KEAPI_CPU_FREQUENCY_TURBO)).
out	*pFrequency*	Frequency of the CPU core, in kHz.

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – wrong *freqType* value
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.4.2 KEApiGetCpuInfo

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetCpuInfo (
      PKEAPI_CPU_INFO pCpuInfo
);
```

Description:

Provides information about processors.

Parameters:

in/out	Parameter name	Description
out	*pCpuInfo*	Returned KEAPI_CPU_INFO structure

Structure used:

```
typedef struct Keapi_Cpu_Info
{
     char         cpuName[KEAPI_MAX_STR];  // CPU name,
                                 // zero length string if not available.
     int32_t      cpuCount;          // Number of CPUs
     int32_t      cpuCoreCount;      // Number of cores of each CPU
     int32_t      cpuThreadCount;    // Number of CPU threads
} KEAPI_CPU_INFO, *PKEAPI_CPU_INFO;
```

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.4.3 KEApiGetCpuPerformance

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetCpuPerformance (
      int32_t coreNr,
      int8_t *pPerformancePercentage
);
```

Description:

Provides information about the current CPU core performance in percentage

Parameters:

in/out	Parameter name	Description
in	*coreNr*	CPU core number (zero based, global counter through all CPUs).
out	*pPerformancePercentage*	Pointer to current CPU core performance in percentage.

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.5 Memory

3.5.1 KEApiGetMemoryInfo

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetMemoryInfo (
      PKEAPI_MEMORY_INFO pMemoryInfo
);
```

Description:

Provides information about physical memory.

Parameters:

in/out	Parameter name	Description
out	*pMemoryInfo*	Returned KEAPI_MEMORY_INFO structure

Structure used:

```
typedef struct Keapi_Memory_Info
{
     int32_t     memTotal;        // Total physical memory size in MB
     int32_t     memFree;         // Free memory in MB
     int32_t     memSpeed;        // Memory speed in MHz
     char        memType[KEAPI_MAX_STR];  // Type of memory (DDR, DDR2, etc.
                                   // zero length string if not available.
} KEAPI_MEMORY_INFO, *PKEAPI_MEMORY_INFO;
```

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.6 Hard disks and mount points

3.6.1 KEApiGetDiskDriveCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetDiskDriveCount (
      int32_t *pDiskDriveCount
);
```

Description:

Provides number of installed disk drives.

Parameters:

in/out	Parameter name	Description
out	*pDiskDriveCount*	Number of installed disk drives

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.6.2 KEApiGetDiskDriveList

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetDiskDriveList (
      PKEAPI_DISK_DRIVE pDiskDrives,
      int32_t diskDriveCount
);
```

Description:

Provides list of disk drives and their properties.

Parameters:

in/out	Parameter name	Description
in	*diskDriveCount*	Number of disks
out	*pDiskDrives*	Returned array of KEAPI_DISK_DRIVE structures. The array must be allocated as `DiskDriveCount * sizeof(KEAPI_DISK_DRIVE)`, where *DiskDriveCount* is obtained from calling **KEApiGetDiskDriveCount**.

Structure used:

```
typedef struct Keapi_Disk_Drive
{
    char        name[KEAPI_MAX_STR];                    // HDD name
    char        model[KEAPI_MAX_STR];                   // Model
    char        diskSerialNumber[KEAPI_MAX_STR];    // Serial number
    int32_t     size;                           // Size in MB
} KEAPI_DISK_DRIVE, *PKEAPI_DISK_DRIVE;
```

Returns:

`KEAPI_RET_SUCCESS`
`KEAPI_RET_NOT_INITIALIZED`
`KEAPI_RET_PARAM_NULL`
`KEAPI_RET_PARAM_ERROR` – parameter `size` value is more than available disks number
`KEAPI_RET_PARTIAL_SUCCESS` – parameter `size` value is less than available disks number
`KEAPI_RET_FUNCTION_NOT_SUPPORTED`
`KEAPI_RET_FUNCTION_NOT_IMPLEMENTED`
`KEAPI_RET_ERROR` – other error

3.6.3 KEApiGetDiskDriveSMARTAttrCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetDiskDriveSMARTAttrCount (
      int32_t diskNr,
      int32_t *pAttrCount
);
```

Description:

Provides number of SMART attributes of the disk drive.

Parameters:

in/out	Parameter name	Description
in	*diskNr*	Disk number from 0 to *diskDriveCount – 1*
out	*pAttrCount*	Where to put number of SMART attributes of the disk drive

Returns:

`KEAPI_RET_SUCCESS`
`KEAPI_RET_NOT_INITIALIZED`
`KEAPI_RET_PARAM_NULL`
`KEAPI_RET_PARAM_ERROR` – wrong *diskNr*
`KEAPI_RET_FUNCTION_NOT_SUPPORTED`
`KEAPI_RET_FUNCTION_NOT_IMPLEMENTED`
`KEAPI_RET_ERROR` – other error

3.6.4 KEApiGetDiskDriveSMARTAttrs

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetDiskSMARTAttrs (
      int32_t diskNr,
      PKEAPI_SMART_ATTR pAttrs,
      int32_t attrCount
);
```

Description:

Provides list of disk SMART attributes.

Parameters:

in/out	Parameter name	Description
in	*diskNr*	
out	*pAttrs*	Array of **KEAPI_SMART_ATTR** structures
in	*attrCount*	Number of attributes (size of the *pAttrs* array)

Structure used:

```
typedef struct Keapi_Smart_Attr
{
     uint8_t        attrID;          // attribute ID
     uint16_t       statusFlags;     //
     uint8_t        attrValue;       // normalized value
     uint8_t        worstValue;      // worst value
     uint8_t        rawValue[6];     // raw value
} KEAPI_SMART_ATTR, *PKEAPI_SMART_ATTR;
```

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR –**attrCount** value is more than available, wrong *diskNr*
KEAPI_RET_PARTIAL_SUCCESS –**attrCount** value is less than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.6.5 KEApiGetMountPointCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetMountPointCount (
      int32_t *pCount
);
```

Description:

Provides number of mount points in the system.

Parameters:

in/out	Parameter name	Description
out	**pCount**	Number of mount points

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.6.6 KEApiGetMountPointList

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetMountPointList (
      PKEAPI_MOUNT_POINT pMountPointList,
      int32_t mountPointCount
);
```

Description:

Provides list of mount points information descriptors.

Parameters:

in/out	Parameter name	Description
in	**mountPointCount**	Number of mount points (size of elements in the **pMountPointList** buffer)
out	**pMountPointList**	Array of **KEAPI_MOUNT_POINT** structures

Structure used:

```
typedef struct Keapi_Mount_Point
{
     char           name[KEAPI_MAX_STR];   // mount point name
     char           fsType[KEAPI_MAX_STR]; // Filesystem type
     int32_t        size;             // Size in MB
     int32_t        freeSpace;        // Free space in MB
} KEAPI_MOUNT_POINT, * PKEAPI_MOUNT_POINT;
```

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – parameter **mountPointCount** value is more than available
KEAPI_RET_PARTIAL_SUCCESS – parameter **attrCount** value is less than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.7 Battery

3.7.1 KEApiGetBatteryCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetBatteryCount (
      int32_t *pBatteryCount
);
```

Description:

Provides number of available battery slots. SMBUS specifies 4 slots.

Parameters:

in/out	Parameter name	Description
out	*pBatteryCount*	Number of available battery slots

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.7.2 KEApiGetBatteryInfo

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetBatteryInfo (
      int32_t slotNr,
      PKEAPI_BATTERY_INFO pBatteryInfo
);
```

Description:

Provides information about battery in slot.

Parameters:

in/out	Parameter name	Description
in	*slotNr*	Requested battery's slot number. SMBUS specifies 4 slots. Numbers start with zero.
out	*pBatteryInfo*	Returned **KEAPI_BATTERY_INFO** structure

Structure used:

```
typedef struct Keapi_Battery_Info

{

     char         deviceName[KEAPI_MAX_STR];     // Device name

     char         type[KEAPI_MAX_STR];   // LION, NiCd, NiMH…

     char         serialNumber[KEAPI_MAX_STR];   // Serial number

     int32_t      designedVoltage;       // Designed voltage in mV

     int32_t      designedCapacity;      // Designed capacity of fully
charged battery in mAh
```

```
    int32_t     fullyChargedCapacity;  // Real capacity of fully charged
battery in mAh
    int32_t     cycleCount;            // Number of charge/discharge cycles
experienced during lifetime
} KEAPI_BATTERY_INFO, *PKEAPI_BATTERY_INFO;
```

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – parameter *slotNr* value is more than available
KEAPI_RET_RETRIEVAL_ERROR – no battery in the slot
KEAPI_RET_PARTIAL_SUCCESS – not all fields are filled
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.7.3 KEApiGetBatteryState

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetBatteryState (
      int32_t slotNr,
      PKEAPI_BATTERY_STATE pBatteryState
);
```

Description:

Provides information about selected battery.

Parameters:

in/out	Parameter name	Description
in	*slotNr*	Requested battery's slot number. Numbers start with zero.
Out	*pBatteryState*	Pointer to a KEAPI_BATTERY_STATE structure

Structure used:

```
typedef struct Keeapi_BatteryState
{
    int32_t     powerState;      // Current power state:
                // charging = 0, charged = 1, discharging = 2
    int32_t     fullBatteryRemainingTime;      // Remaining time in seconds
                // when battery is full and AC power unplugged
    int32_t     remainingTime;   // Remaining time in seconds
    int32_t     remainingCapacity; // Remaining capacity in mAh
    int32_t     currentVoltage;  // Current voltage in mV
    int32_t     rate;            // Current charging/discharging rate in mA
    int32_t     chargeState;     // Battery charge state in percentage
} KEAPI_BATTERY_STATE, *PKEAPI_BATTERY_STATE;
```

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED

KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – parameter *slotNr* value is more than available
KEAPI_RET_RETRIEVAL_ERROR – no battery in the slot
KEAPI_RET_PARTIAL_SUCCESS – not all fields are filled
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.8 Performance

3.8.1 KEApiPerformanceStateCaps

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiPerformanceStateCaps (
      uint32_t *pStatesMask
);
```

Description:

Provides power states mask the system is capable to run.

Parameters:

in/out	Parameter name	Description
out	*pStatesMask*	Pointer to states mask returned, available masks: KEAPI_PM_P0 – max power and frequency (always capable) KEAPI_PM_P1 – less than P0, voltage/frequency scaled KEAPI_PM_P2 KEAPI_PM_P3 … KEAPI_PM_P16 – less than P15, voltage/frequency scaled

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.8.2 KEApiGetPerformanceStateDescription

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetPerformanceStateDescription (
      uint32_t state,
      char *pDescription
);
```

Description:

Get description of specified power state.

Parameters:

in/out	Parameter name	Description
in	*state*	the state to be described: KEAPI_PM_P0 KEAPI_PM_P1 KEAPI_PM_P2 KEAPI_PM_P3 … KEAPI_PM_P16
out	*pDescription*	buffer with KEAPI_MAX_STR capacity to put state description in arbitrary text form

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – parameter *state* value is more than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.8.3 KEApiGetPerformanceState

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetPerformanceState (
      uint32_t *pState
);
```

Description:

Get information about device power state.

Parameters:

in/out	Parameter name	Description
out	*pState*	Pointer to the state value returned: KEAPI_PM_P0 KEAPI_PM_P1 KEAPI_PM_P2 KEAPI_PM_P3 … KEAPI_PM_P16

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.8.4 KEApiSetPerformanceState

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSetPerformanceState (
      uint32_t state
);
```

Description:

Prepares system to enter to defined performance state.

Parameters:

in/out	Parameter name	Description
in	*state*	the state requested: KEAPI_PM_P0 KEAPI_PM_P1 KEAPI_PM_P2 KEAPI_PM_P3 …

		KEAPI_PM_P16

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_ERROR –*state* value is not supported
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.9 Temperature sensors

3.9.1 KEApiGetTempSensorCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetTempSensorCount (
      int32_t *pTempSensorCount
);
```

Description:

Provides number of temperature sensors.

Parameters:

in/out	Parameter name	Description
out	*pTempSensorCount*	Pointer to number of installed temperature sensors

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.9.2 KEApiGetTempSensorValue

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetTempSensorValue (
      int32_t sensorNr,
      PKEAPI_SENSOR_VALUE pSensorValue
);
```

Description:

Derives information about current value of a temperature sensor with a given ID.

Parameters:

in/out	Parameter name	Description
in	*sensorNr*	Number (index) of a temperature sensor. Numbers start with 0
out	*pSensorValue*	Pointer to the value structure to store sensor status and value (in 1/1000 Celsius, negative values allowed)

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – *sensorNr* value is more than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

Structure used:

```
typedef struct Keapi_Sensor_Value
{
 int32_t value; // Value obtained from sensor (negative values allowed)
 int32_t status; // Sensor’s status:
//          KEAPI_SENSOR_STATUS_ACTIVE /* Sensor is operating  */
//          KEAPI_SENSOR_STATUS_ALARM /* Sensor reports alarm condition */
//          KEAPI_SENSOR_STATUS_BROKEN /* Sensor circuit is broken */
//    KEAPI_SENSOR_STATUS_SHORTCIRCUIT /* Sensor has a short circuit */
} KEAPI_SENSOR_VALUE, * PKEAPI_SENSOR_VALUE;
```

3.9.3 KEApiGetTempSensorValueList

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetTempSensorValueList (
      PKEAPI_SENSOR_VALUE pSensorValues,
      int32_t sensorCount
);
```

Description:

Provides information about temperature sensors (current value and status), each stored in the KEAPI_SENSOR_VALUE structure

Parameters:

in/out	Parameter name	Description
out	*pSensorValues*	Buffer to store value list
in	*sensorCount*	Number of temperature sensors (size of buffer / sizeof(KEAPI_SENSOR_VALUE))

Structure used: See KEApiGetTempSensorValue

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – *sensorCount* value is more than available
KEAPI_RET_PARTIAL_SUCCESS – *sensorCount* value is less than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.9.4 KEApiGetTempSensorInfo

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetTempSensorInfo (
      int32_t sensorNr,
      PKEAPI_SENSOR_INFO pSensorInfo
);
```

Description:

Derives detailed information sensor settings.

Parameters:

in/out	Parameter name	Description
in	*sensorNr*	Number (index) of a temperature sensor. Numbers start with 0

out	*pSensorInfo*	Buffer to store the info. Values are in 1/1000 Celsius (negative values allowed)

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – *sensorNr* value is more than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

Structure used: (fields which do not provide valid information are set to KEAPI_SENSOR_INFO_UNKNOWN)

```
typedef struct Keapi_Sensor_Info
{
 char name[KEAPI_MAX_STR]; // Sensor's description
 int32_t type; // Sensor’s type (see values below)
 int32_t min; // minimum sensor value that can be measured
 int32_t max; // minimum sensor value that can be measured
 int32_t alarmHi; // upper alarm threshold,
// i.e. the value up to which the value must rise to generate an alarm
 int32_t hystHi; // upper alarm hysteresis,
            // i.e. how much the value must decrease to reset an alarm,
            // must be reported as an absolute value, NOT a delta
 int32_t alarmLo; // lower alarm threshold
 int32_t hystLo; // lower alarm hysteresis
} KEAPI_SENSOR_INFO, * PKEAPI_SENSOR_INFO;
```

Sensor types:

KEAPI_TEMP_CPU

KEAPI_TEMP_BOX

KEAPI_TEMP_ENV

KEAPI_TEMP_BOARD

KEAPI_TEMP_BACKPLANE

KEAPI_TEMP_CHIPSET

KEAPI_TEMP_VIDEO

KEAPI_TEMP_OTHER

3.10 Voltage sensors

3.10.1 KEApiGetVoltageSensorCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetVoltageSensorCount (
      int32_t *pVoltageSensorCount
);
```

Description:

Provides number of Voltage sensors.

Parameters:

in/out	Parameter name	Description
out	*pVoltageSensorCount*	Pointer to number of installed Voltage sensors

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.10.2 KEApiGetVoltageSensorValue

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetVoltageSensorValue (
      int32_t sensorNr,
      PKEAPI_SENSOR_VALUE pSensorValue
);
```

Description:

Derives information about current value of a Voltage sensor with a given ID.

Parameters:

in/out	Parameter name	Description
in	*sensorNr*	Number (index) of a sensor. Numbers start with 0
out	*pSensorValue*	Pointer to the value structure to store sensor status and value (in 1/1000 Volts)

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – *sensorNr* value is more than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

Structure used: See KEApiGetTempSensorValue

3.10.3 KEApiGetVoltageSensorValueList

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetVoltageSensorValueList (
      PKEAPI_SENSOR_VALUE pSensorValues,
      int32_t sensorCount
);
```

Description:

Provides information about Voltage sensors (current value and status), each stored in the KEAPI_SENSOR_VALUE structure

Parameters:

in/out	Parameter name	Description
out	*pSensorValues*	Buffer to store value list
in	*sensorCount*	Number of Voltage sensors (size of buffer / sizeof(KEAPI_SENSOR_VALUE))

Structure used: See KEApiGetTempSensorValue

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR –*sensorCount* value is more than available
KEAPI_RET_PARTIAL_SUCCESS –*sensorCount* value is less than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.10.4 KEApiGetVoltageSensorInfo

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetVoltageSensorInfo (
      int32_t sensorNr,
      PKEAPI_SENSOR_INFO pSensorInfo
);
```

Description:

Derives detailed information sensor settings.

Parameters:

in/out	Parameter name	Description
in	*sensorNr*	Number (index) of a Voltage sensor. Numbers start with 0
out	*pSensorInfo*	Buffer to store the info. Values are in 1/1000 Volts

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – *sensorNr* value is more than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

Structure used: See KEApiGetTempSensorInfo

Sensor types:

KEAPI_VOLTAGE_VCORE

KEAPI_VOLTAGE_1V8

KEAPI_VOLTAGE_2V5

KEAPI_VOLTAGE_3V3

KEAPI_VOLTAGE_VBAT

KEAPI_VOLTAGE_5V

KEAPI_VOLTAGE_5VSB

KEAPI_VOLTAGE_12V

KEAPI_VOLTAGE_AC

KEAPI_VOLTAGE_OTHER

3.11 Fan Sensors

3.11.1 KEApiGetFanSensorCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetFanSensorCount (
      int32_t *pFanSensorCount
);
```

Description:

Provides number of fans.

Parameters:

in/out	Parameter name	Description
out	*pFanSensorCount*	Pointer to number of fans

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.11.2 KEApiGetFanSensorValue

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetFanSensorValue (
      int32_t sensorNr,
      PKEAPI_SENSOR_VALUE pSensorValue
);
```

Description:

Derives information about current value of a Fan sensor with a given ID.

Parameters:

in/out	Parameter name	Description
in	*sensorNr*	Number (index) of a sensor. Numbers start with 0
out	*pSensorValue*	Pointer to the value structure to store sensor status and value (in RPMs - revolutions-per-minute))

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – *sensorNr* value is more than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

Structure used: See KEApiGetTempSensorValue

3.11.3 KEApiGetFanSensorValueList

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetFanSensorValueList (
      PKEAPI_SENSOR_VALUE pSensorValues,
      int32_t sensorCount
);
```

Description:

Provides information about Fan sensors (current value and status), each stored in the KEAPI_SENSOR_VALUE structure

Parameters:

in/out	Parameter name	Description
out	*pSensorValues*	Buffer to store value list
in	*sensorCount*	Number of Fan sensors (size of buffer / sizeof(KEAPI_SENSOR_VALUE))

Structure used: See KEApiGetTempSensorValue

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR –*sensorCount* value is more than available
KEAPI_RET_PARTIAL_SUCCESS –*sensorCount* value is less than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.11.4 KEApiGetFanSensorInfo

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetFanSensorInfo (
      int32_t sensorNr,
      PKEAPI_SENSOR_INFO pSensorInfo
);
```

Description:

Derives detailed information sensor settings.

Parameters:

in/out	Parameter name	Description
in	*sensorNr*	Number (index) of a Fan sensor. Numbers start with 0
out	*pSensorInfo*	Buffer to store the info, values are in RPMs (revolutions-per-minute)

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – *sensorNr* value is more than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

Structure used: See KEApiGetTempSensorInfo

Sensor types:

KEAPI_FAN_CPU

KEAPI_FAN_BOX

KEAPI_FAN_ENV

KEAPI_FAN_CHIPSET

KEAPI_FAN_VIDEO

KEAPI_FAN_OTHER

3.12 Display

3.12.1 KEApiGetDisplayCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetDisplayCount (
      int32_t *pDisplayCount
);
```

Description:

Provides number of installed displays.

Parameters:

in/out	Parameter name	Description
out	*pDisplayCount*	Number of installed displays

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.12.2 KEApiGetBacklightValue

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetBacklightValue (
      int32_t displayNr,
      int32_t *pValue
);
```

Description:

Provides information about current backlight intensity of the selected display.

Parameters:

in/out	Parameter name	Description
in	*displayNr*	Requested display's number. Numbers start with zero
out	*pValue*	Pointer to variable that receives actual brightness intensity. The value ranges from 0 to `KEAPI_DISPLAY_BRIGHTNESS_MAX` (255).

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such *displayNr*
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.12.3 KEApiSetBacklightValue

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSetBacklightValue (
      int32_t displayNr,
        int32_t value
);
```

Description:

Enables backlight and sets backlight intensity of the selected display. Value 0 sets backlight to OFF.

Parameters:

in/out	Parameter name	Description
in	*displayNr*	Requested display's number. Numbers start with zero
out	*value*	Backlight intensity. The value ranges from 0 to `KEAPI_DISPLAY_BRIGHTNESS_MAX` (255). Value 0 sets backlight to OFF.

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_ERROR – no such *displayNr* , value is not in allowed range
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.13 Network and PCI devices

3.13.1 KEApiGetNetworkDeviceCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetNetworkDeviceCount (
      int32_t *pNetworkDeviceCount
);
```

Description:

Provides a number of installed network devices.

Parameters:

in/out	Parameter name	Description
out	*pNetworkDeviceCount*	Number of installed network devices

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.13.2 KEApiGetNetworkDeviceList

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetNetworkDeviceList (
      PKEAPI_NETWORK_DEVICE pNetworkDevices,
      int32_t networkDeviceCount
);
```

Description:

Provides information about installed network devices.

Parameters:

in/out	Parameter name	Description
in	*networkDeviceCount*	Number of network devices
out	*pNetworkDevices*	Pointer KEAPI_NETWORK_DEVICE list buffer. The array must be preallocated as *NetworkDeviceCount* * sizeof(KEAPI_NETWORK_DEVICE).

Structure used:

```
typedef struct Keapi_Network_Device

{

     char         ip[KEAPI_MAX_STR]; // IP address

     char         mac[KEAPI_MAX_STR];       // MAC address in format XX-XX-XX-
XX-XX-XX

     int32_t      speed;            // Connection speed in MB/s

     char         deviceName[KEAPI_MAX_STR];     // Name of the network
device
} KEAPI_NETWORK_DEVICE, *PKEAPI_NETWORK_DEVICE;
```

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – *networkDeviceCount* value is more than available or < 0
KEAPI_RET_PARTIAL_SUCCESS – *networkDeviceCount* value is less than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.13.3 KEApiGetPciDeviceCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetPciDeviceCount (
      int32_t *pPciDeviceCount
);
```

Description:

Provides a number of installed PCI devices.

Parameters:

in/out	Parameter name	Description
out	*pPciDeviceCount*	Number of installed PCI devices

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.13.4 KEApiGetPciDeviceList

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetPciDeviceList (
      PKEAPI_PCI_DEVICE pPciDevices,
      int32_t pciDeviceCount
);
```

Description:

Provides a list of PCI devices.

Parameters:

in/out	Parameter name	Description
in	*pciDeviceCount*	Number of installed PCI devices
out	*pPciDevices*	Returned array of KEAPI_PCI_DEVICE structures

Structure used:

```
typedef struct Keapi_Pci_Device

{

     int32_t      domain;                 // Domain number
     int32_t     bus;                    // Bus number
     int32_t     slot;                   // Slot number
     int32_t     funct;                  // Function number
     int32_t     deviceId;               // Device ID
     int32_t     vendorId;               // Vendor ID
     int32_t     classId;                // Class ID
     char        deviceName[KEAPI_MAX_STR];     // Name of the device
     char        vendorName[KEAPI_MAX_STR];     // Name of the vendor
     char        className[KEAPI_MAX_STR];      // Name of the class
} KEAPI_PCI_DEVICE, *PKEAPI_PCI_DEVICE;
```

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – *pciDeviceCount* value is more than available or < 0
KEAPI_RET_PARTIAL_SUCCESS – *pciDeviceCount* value is less than available
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.14 Storage area

3.14.1 KEApiGetStorageCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetStorageCount (
      int32_t *pStorageCount
);
```

Description:

Provides number of EEPROM storage areas.

Parameters:

in/out	Parameter name	Description
out	*pStorageCount*	Number of available storage areas

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.14.2 KEApiGetStorageSize

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetStorageSize (
      int32_t storageNr,
      int32_t *pStorageSize
);
```

Description:

Provides information about EEPROM storage area's size.

Parameters:

in/out	Parameter name	Description
in	*storageNr*	Number of the storage area (starting from 0).
out	*pStorageSize*	Pointer to variable that receives size of the selected storage area.

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such *storageNr*
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.14.3 KEApiStorageRead

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiStorageRead (
      int32_t storageNr,
      int32_t offset,
      uint8_t *pData,
      int32_t dataLength
);
```

Description:

Reads block of bytes from selected EEPROM.

Parameters:

in/out	Parameter name	Description
in	*storageNr*	EEPROM storage number (starts from 0).
in	*offset*	Start address offset
out	*pData*	Pointer to buffer that receives data
in	*dataLength*	Number of bytes to read

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such *storageNr*
KEAPI_RET_READ_ERROR
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.14.4 KEApiStorageWrite

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiStorageWrite (
      int32_t storageNr,
      int32_t offset,
      uint8_t *pData,
      int32_t dataLength
);
```

Description:

Writes block of bytes to selected EEPROM.

Parameters:

in/out	Parameter name	Description
in	*storageNr*	EEPROM storage number (enumerated from 0).
in	*offset*	Start address offset
in	*pData*	Pointer to buffer that contains data to write to EEPROM
in	*dataLength*	Number of bytes to write

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such *storageNr*
KEAPI_RET_WRITE_ERROR
KEAPI_RET_FUNCTION_NOT_SUPPORTED

`KEAPI_RET_FUNCTION_NOT_IMPLEMENTED`
`KEAPI_RET_ERROR` – other error

3.15 I2C

The I2C specification defines a 7 bit and a 10 bit address format. Only 7 bit addresses are allowed in i2cAddress parameter for KEAPI function. This is because 10 Bit addresses are realized in the I2C Specification as an extended write read transfer thus can be addressable as 7 Bit devices.

3.15.1 KEApiGetI2cBusCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetI2cBusCount (
      int32_t *pI2cBusCount
);
```

Description:

Function for getting number of active I2C buses. Some hardware types reserves specific bus numbers to specific types (see definitions in keapi.h, eapi.h, Jida.h):

```
      EAPI_ID_I2C_EXTERNAL
      EAPI_ID_I2C_LVDS_1
      EAPI_ID_I2C_LVDS_2
      JIDA_I2C_TYPE_PRIMARY
      JIDA_I2C_TYPE_JILI
```

Parameters:

in/out	Parameter name	Description
Out	*pI2cBusCount*	Pointer to the variable where the I2C bus count is saved

Returns:

```
KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_BUS_ERROR
KEAPI_RET_CANCELLED
KEAPI_RET_BUSY_COLLISION
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
```
`KEAPI_RET_ERROR` – other error

3.15.2 KEApiI2cXfer

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiI2cXfer (
      int32_t i2cNr,
      uint8_t i2cAddress,
      uint8_t *pWriteData,
      int32_t writeLength,
      uint8_t *pReadData,
      int32_t *pReadLength
);
```

Description:

Universal function for write-read operations to the I2C bus. This function performs write operation passing device address and writes data, then performs I2C START, transfer device address and reads data from the slave I2C device connected to the I2C bus. Write operation will not be performed if no write data is provided. The I2C operations sequence shall be:

```
[Start<Addr Byte><W>Ack<Write Data Byte[1]>Ack
 … < Write Data Byte[writeLength]>Ack]
Start<Addr Byte><R>Ack<Read Data Byte[1]>Ack
 … <Read Data Byte[readLength]>Nak Stop
```

Parameters:

in/out	Parameter name	Description
In	*i2cNr*	Number of I2C Bus. From 0 to (*I2cBusCount* – 1) returned by **KEApiGetI2cBusCount**.
In	*i2cAddress*	Address of I2C slave device
in	*pWriteData*	Data to write, can be `NULL` if `writeLength == 0`
in	*writeLength*	Length of data to write
out	*pReadData*	Buffer for read data, can be `NULL` if *pReadLength* `== 0`
inout	*pReadLength*	Also an “out” parameter. When the function finishes, this parameter contains a real value of the read data length.

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such *i2cNr*
KEAPI_RET_READ_ERROR
KEAPI_RET_WRITE_ERROR
KEAPI_RET_BUS_ERROR
KEAPI_RET_CANCELLED
KEAPI_RET_BUSY_COLLISION
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.15.3 KEApiI2cProbe

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiI2cProbe (
      int32_t i2cNr,
      uint8_t i2cAddress,
      uint8_t memoryAddress,
      uint8_t memoryAddressUsed
);
```

Description:

Probes I2C Device. There are two methods to probe I2C device: 1-st is to perform write operation:

```
Start<Addr Byte><W>Ack<Memory Address Byte>Ack Stop
```

This sequence can be used to probe a specific register of I2C device, or to implement 10-bit addressing on I2C bus.

Another method is to perform only device probe:

```
Start<Addr Byte><W>Ack Stop
```

This sequence is performing by KEAPI when `memoryAddressUsed` is `FALSE`.

Parameters:

in/out	Parameter name	Description
In	*i2cNr*	Number of I2C Bus. From 0 to (*I2cBusCount* – 1) returned by **KEApiGetI2cBusCount**.
In	*i2cAddress*	Address of I2C slave device
in	*memoryAddress*	Address of register/memory (for 10-bit I2C addressing or register address inside I2C device).
In	*memoryAddressUsed*	If not 0 – memoryAddress byte data has to be written to I2C.

Returns:

`KEAPI_RET_SUCCESS` – probe success
`KEAPI_RET_NOT_INITIALIZED`
`KEAPI_RET_PARAM_ERROR` – no such *i2cNr*
`KEAPI_RET_BUS_ERROR` – no such device
`KEAPI_RET_CANCELLED`
`KEAPI_RET_BUSY_COLLISION`
`KEAPI_RET_FUNCTION_NOT_SUPPORTED`
`KEAPI_RET_FUNCTION_NOT_IMPLEMENTED`
`KEAPI_RET_ERROR` – other error

3.16 SPI

3.16.1 KEApiGetSpiBusCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetSpiBusCount (
      int32_t *pSpiBusCount
);
```

Description:

Function for getting number of active SPI Buses.

Parameters:

in/out	Parameter name	Description
out	*pSpiBusCount*	Pointer to variable to save SpiBus count

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.16.2 KEApiSpiXfer

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSpiXfer (
      int32_t spiNr,
      uint16_t deviceId,
      uint32_t command,
      uint8_t commandSize,
      int32_t numTransfers,
      uint8_t *pWriteData,
      uint8_t *pReadData
);
```

Description:

Exchange SPI data. This function transfers data between the master SPI bus controller and a slave SPI device. Each transfer consists of a command word, receive data and transmit data. The command format is device specific. It typically directs I/O to a specific register set within the device and may identify a data transfer direction (input/output). If the transmit buffer is null, zeros are sent out to the slave. If the receive buffer is null, no data is read.

The transfer starts by asserting the chip select code as defined in the deviceId. The command word is then transmitted to the slave, command word size is specified in bytes, 0 value in `commandSize` defines no use of command word. Data provided in the transmit buffer is sent out on the SPI MOSI line and receive data provided by the slave by MISO line is captured in the receive buffer. The function returns when the number of payload transfers have completed.

Parameters:

in/out	Parameter name	Description

In	*spiNr*	I2C Bus id. From 0 to (spi*BusCount* – 1) returned by **KEApiGetSpiBusCount**.
In	*deviceId*	device number (chip select)
in	*command*	command word
in	*commandSize*	command word size in bytes (lowest bits of 32-bit word are used)
in	*numTransfers*	total number of transfers (write-read byte transaction) excluding command
in	*pWriteData*	data to write (of *numTransfers* size), NULL means read-only operation
out	*pReadData*	read data buffer (of *numTransfers* size), NULL means write-only operation

Returns:

`KEAPI_RET_SUCCESS`
`KEAPI_RET_NOT_INITIALIZED`
`KEAPI_RET_PARAM_NULL`
`KEAPI_RET_PARAM_ERROR` – no such `spiNr`
`KEAPI_RET_READ_ERROR`
`KEAPI_RET_WRITE_ERROR`
`KEAPI_RET_FUNCTION_NOT_SUPPORTED`
`KEAPI_RET_FUNCTION_NOT_IMPLEMENTED`
`KEAPI_RET_ERROR` – other error

3.17 SMBus

3.17.1 KEApiGetSmbusCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetSmbusCount (
        int32_t *pSmbusCount
);
```

Description:

Function for getting number of active SMBuses.

Parameters:

in/out	Parameter name	Description
out	*pSmbusCount*	Pointer to variable to save SMBus count

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.17.2 KEApiSmbusQuickCommand

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSmbusQuickCommand (
        int32_t smbusNr,
        uint8_t smbusAddress,
        uint8_t operation
);
```

Description:

Quick command read/write may be used to simply turn a device on/off or to enable/disable low-power standby mode etc. There is no data received. For additional information, refer to the System Management Bus(SMBus) Specification Version 2.0, which is available at http://smbus.org/specs/smbus20.pdf.

Parameters:

in/out	Parameter name	Description
In	*smbusNr*	Number of SMBus. From 0 to (*SmbusCount* – 1) returned by **KEApiGetSmbusCount**.
In	*smbusAddress*	Address of SMBus slave device
in	*operation*	What should be done • KEAPI_SMBUS_WRITE_OP (0) • KEAPI_SMBUS_READ_OP (1)

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED

`KEAPI_RET_PARAM_ERROR` – no such *smbusNr*, wrong operation
`KEAPI_RET_BUS_ERROR`
`KEAPI_RET_CANCELLED`
`KEAPI_RET_BUSY_COLLISION`
`KEAPI_RET_FUNCTION_NOT_SUPPORTED`
`KEAPI_RET_FUNCTION_NOT_IMPLEMENTED`
`KEAPI_RET_ERROR` – other error

3.17.3 KEApiSmbusSendByte

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSmbusSendByte (
        int32_t smbusNr,
        uint8_t smbusAddress,
        uint8_t byte
);
```

Description:

A simple device may accept up to 256 possible encoded commands in a form of a byte. For additional information, refer to the System Management Bus (SMBus) Specification Version 2.0, which is available at http://smbus.org/specs/smbus20.pdf.

Parameters:

in/out	Parameter name	Description
In	*smbusNr*	Number of SMBus. From 0 to (*SmbusCount* – 1) returned by **KEApiGetSmbusCount**.
In	*smbusAddress*	Address of SMBus slave device
in	*byte*	Command. Depends on device

Returns:

`KEAPI_RET_SUCCESS`
`KEAPI_RET_NOT_INITIALIZED`
`KEAPI_RET_PARAM_ERROR` – no such *smbusNr*
`KEAPI_RET_BUS_ERROR`
`KEAPI_RET_CANCELLED`
`KEAPI_RET_BUSY_COLLISION`
`KEAPI_RET_FUNCTION_NOT_SUPPORTED`
`KEAPI_RET_FUNCTION_NOT_IMPLEMENTED`
`KEAPI_RET_ERROR` – other error

3.17.4 KEApiSmbusReceiveByte

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSmbusReceiveByte (
        int32_t smbusNr,
        uint8_t smbusAddress,
        uint8_t *pByte
);
```

Description:

A simple device may have information that the host needs. It can do so with Receive byte protocol. For additional information, refer to the System Management Bus (SMBus) Specification Version 2.0, which is available at http://smbus.org/specs/smbus20.pdf.

Parameters:

in/out	Parameter name	Description
In	*smbusNr*	Number of SMBus. From 0 to (*SmbusCount* – 1) returned by **KEApiGetSmbusCount**.
In	*smbusAddress*	Address of SMBus slave device
out	*pByte*	Device information byte. Depends on device

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such *smbusNr*
KEAPI_RET_BUS_ERROR
KEAPI_RET_CANCELLED
KEAPI_RET_BUSY_COLLISION
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.17.5 KEApiSmbusWriteByte

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSmbusWriteByte (
        int32_t smbusNr,
        uint8_t smbusAddress,
        uint8_t command,
        uint8_t byte
);
```

Description:

This function writes data of size of byte to a device. For additional information, refer to the System Management Bus (SMBus) Specification Version 2.0, which is available at http://smbus.org/specs/smbus20.pdf.

Parameters:

in/out	Parameter name	Description
In	*smbusNr*	Number of SMBus. From 0 to (*SmbusCount* – 1) returned by **KEApiGetSmbusCount**.
In	*smbusAddress*	Address of SMBus slave device
in	*command*	Command code. Depends on device
in	*byte*	Data

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_ERROR – no such *smbusNr*
KEAPI_RET_BUS_ERROR
KEAPI_RET_CANCELLED
KEAPI_RET_BUSY_COLLISION
KEAPI_RET_FUNCTION_NOT_SUPPORTED

KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.17.6 KEApiSmbusReadByte

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSmbusReadByte (
        int32_t smbusNr,
        uint8_t smbusAddress,
        uint8_t command,
        uint8_t *pByte
);
```

Description:

This function reads data of size of byte from a device. For additional information, refer to the System Management Bus (SMBus) Specification Version 2.0, which is available at http://smbus.org/specs/smbus20.pdf.

Parameters:

in/out	Parameter name	Description
In	*smbusNr*	Number of SMBus. From 0 to (*SmbusCount* – 1) returned by **KEApiGetSmbusCount**.
In	*smbusAddress*	Address of SMBus slave device
in	*command*	Command code. Depends on device
out	*pByte*	Pointer to the data value

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such *smbusNr*
KEAPI_RET_BUS_ERROR
KEAPI_RET_CANCELLED
KEAPI_RET_BUSY_COLLISION
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.17.7 KEApiSmbusWriteWord

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSmbusWriteWord (
        int32_t smbusNr,
        uint8_t smbusAddress,
        uint8_t command,
        uint16_t word
);
```

Description:

This function writes data of size of word to a device. For additional information, refer to the System Management Bus (SMBus) Specification Version 2.0, which is available at http://smbus.org/specs/smbus20.pdf.

Parameters:

in/out	Parameter name	Description
In	*smbusNr*	Number of SMBus. From 0 to (*SmbusCount* – 1) returned by **KEApiGetSmbusCount**.
In	*smbusAddress*	Address of SMBus slave device
in	*command*	Command code. Depends on device
in	*word*	Data word to write

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_ERROR – no such *smbusNr*
KEAPI_RET_BUS_ERROR
KEAPI_RET_CANCELLED
KEAPI_RET_BUSY_COLLISION
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.17.8 KEApiSmbusReadWord

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSmbusReadWord (
        int32_t smbusNr,
        uint8_t smbusAddress,
        uint8_t command,
        uint16_t *pWord
);
```

Description:

This function reads data of size of word from a device. For additional information, refer to the System Management Bus (SMBus) Specification Version 2.0, which is available at http://smbus.org/specs/smbus20.pdf.

Parameters:

in/out	Parameter name	Description
In	*smbusNr*	Number of SMBus. From 0 to (*SmbusCount* – 1) returned by **KEApiGetSmbusCount**.
In	*smbusAddress*	Address of SMBus slave device
in	*command*	Command code. Depends on device
out	*pWord*	Pointer to the data value

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such *smbusNr*
KEAPI_RET_BUS_ERROR
KEAPI_RET_CANCELLED
KEAPI_RET_BUSY_COLLISION
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.17.9 KEApiSmbusWriteBlock

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSmbusWriteBlock (
        int32_t smbusNr,
        uint8_t smbusAddress,
        uint8_t command,
        uint8_t *pData,
        int8_t dataLength
);
```

Description:

This function writes up to 32 bytes to the device. For additional information, refer to the System Management Bus (SMBus) Specification Version 2.0, which is available at http://smbus.org/specs/smbus20.pdf.

Parameters:

in/out	Parameter name	Description
In	*smbusNr*	Number of SMBus. From 0 to (*SmbusCount* – 1) returned by **KEApiGetSmbusCount**.
In	*smbusAddress*	Address of SMBus slave device
in	*command*	Command code. Depends on device
in	*pData*	Pointer to a data block of size up to 32 bytes
in	*dataLength*	Length of a data block

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such *smbusNr*
KEAPI_RET_BUS_ERROR
KEAPI_RET_CANCELLED
KEAPI_RET_BUSY_COLLISION
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.17.10 KEApiSmbusReadBlock

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSmbusReadBlock (
        int32_t smbusNr,
        uint8_t smbusAddress,
        uint8_t command,
        uint8_t *pData,
        int8_t *pDataLength
);
```

Description:

This function reads up to 32 byte from the device. For additional information, refer to the System Management Bus (SMBus) Specification Version 2.0, which is available at http://smbus.org/specs/smbus20.pdf.

Parameters:

in/out	**Parameter name**	**Description**
In	*smbusNr*	Number of SMBus. From 0 to (*SmbusCount* – 1) returned by **KEApiGetSmbusCount**.
In	*smbusAddress*	Address of SMBus slave device
in	*command*	Command code. Depends on device
out	*pData*	Pointer to a data block of size up to 32 bytes
in/ out	*pDataLength*	This is also the “out” parameter. It is a pointer to the length of data to read. After completing the function, this parameter contains real value of the data length.

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such *smbusNr*
KEAPI_RET_BUS_ERROR
KEAPI_RET_CANCELLED
KEAPI_RET_BUSY_COLLISION
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.18 GPIO

3.18.1 KEApiGetGpioPortCount

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetGpioPortCount (
      int32_t *pCount
);
```

Description:

Function for getting number of active GPIO Ports (each GPIO port can contain from 1 to 32 pins accessible simultaneously).

Parameters:

in/out	Parameter name	Description
Out	*pCount*	Pointer to a variable where the GPIO ports count is saved

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.18.2 KEApiGetGpioPortDirectionCaps

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetGpioPortDirectionCaps (
      int32_t portNr,
      uint32_t *pIns,
      uint32_t *pOuts,
);
```

Description:

Get possible pin directions of the GPIO port.

Parameters:

in/out	Parameter name	Description
In	*portNr*	The GPIO port number (from 0 to (port count – 1)).
Out	*pIns*	Pointer to the location that will receive the pins that are inputs. A 1 indicates a pin in the corresponding bit position is capable of being an input.
Out	*pOuts*	Pointer to the location that will receive the pins that are outputs. A 1 indicates a pin in the corresponding bit position is capable of being an output.

Returns:

```
KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such portNr
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error
```

3.18.3 KEApiGetGpioPortDirections

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetGpioPortDirections(
      int32_t portNr,
      uint32_t *pDirections
);
```

Description:

Function for getting current directions of selected GPIO pins.

Parameters:

in/out	Parameter name	Description
In	portNr	The GPIO port number (from 0 to (port count – 1)).
Out	*pDirections*	Pointer to the location that will receive the current direction of the port pins. A 0 bit indicates an OUTPUT, a 1 bit indicates an INPUT pin in the corresponding bit position.

Returns:

```
KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_PARAM_ERROR – no such portNr
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error
```

3.18.4 KEApiSetGpioPortDirections

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSetGpioPortDirections(
      int32_t portNr,
      uint32_t directions
);
```

Description:

Function for setting direction of selected GPIO pin.

Parameters:

in/out	Parameter name	Description
In	*portNr*	The GPIO port number (from 0 to (port count – 1)).

In	*directions*	Direction of the port pins. A 0 bit indicates an OUTPUT, a 1 bit indicates an INPUT pin in the corresponding bit position.

Returns:

`KEAPI_RET_SUCCESS`
`KEAPI_RET_NOT_INITIALIZED`
`KEAPI_RET_PARAM_ERROR` – no such *portNr*
`KEAPI_RET_FUNCTION_NOT_SUPPORTED`
`KEAPI_RET_FUNCTION_NOT_IMPLEMENTED`
`KEAPI_RET_ERROR` – other error

3.18.5 KEApiGetGpioPortLevels

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiGetGpioPortLevels(
      int32_t portNr,
      uint32_t *pLevels
);
```

Description:

Function for getting level of selected GPIO pin.

Parameters:

in/out	Parameter name	Description
In	*portNr*	The GPIO port number (from 0 to (port count – 1)).
Out	*pLevels*	Reads the current state of the IO Port pins. This includes the input and output values.

Returns:

`KEAPI_RET_SUCCESS`
`KEAPI_RET_NOT_INITIALIZED`
`KEAPI_RET_PARAM_NULL`
`KEAPI_RET_PARAM_ERROR` – no such *portNr*
`KEAPI_RET_FUNCTION_NOT_SUPPORTED`
`KEAPI_RET_FUNCTION_NOT_IMPLEMENTED`
`KEAPI_RET_ERROR` – other error

3.18.6 KEApiSetGpioPortLevels

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiSetGpioPortLevels (
      int32_t portNr,
      uint32_t levels
);
```

Description:

Function for setting level of selected GPIO pin.

Parameters:

in/out	Parameter name	Description

In	*portNr*	The GPIO port number (from 0 to (port count – 1)).
In	*levels*	Writes to the output pins of the IO Port.

Returns:

`KEAPI_RET_SUCCESS`
`KEAPI_RET_NOT_INITIALIZED`
`KEAPI_RET_PARAM_ERROR` – no such *portNr*
`KEAPI_RET_FUNCTION_NOT_SUPPORTED`
`KEAPI_RET_FUNCTION_NOT_IMPLEMENTED`
`KEAPI_RET_ERROR` – other error

3.19 Watchdog

KEAPI-enabled systems can support extended watchdog hardware. The watchdog hardware can function in different modes:

- **Reset Mode:**
  The most common operation mode is to generate a hard reset signal in case the watchdog timeout period has expired.
- **Interrupt Mode:**
  In this mode an expired watchdog timer generates an interrupt signal to the system. Watchdog driver is responsible to process the interrupt and send notification to KEAPI application. The notification mechanism is different between Operating Systems but KEAPI provides unified way to have application notified.
- **Timer-Only Mode:**
  In this Mode the watchdog timer expiration does not generate any hardware signal directly but rise the corresponding WTE status (see below). Application can use this mode to work with watchdog in a polling mode.

Multiple stages support

KEAPI provides support for watchdog hardware implementation which have multiple stages. Watchdog stages are executed consequently and can be configured independently. Watchdog trigger action should start the first stage again.

Each stage acts as a timer with its own timeout period and mode. For example the first stage will run in *Interrupt Mode* and second stage starts next timeout period which will reset the system (*Reset Mode*).

Before the watchdog starts all stages has to be configured via KEApiWatchdogSetup API call.

WTE Status

Some systems implement a readable watchdog timer expired status (WTE). This status can be read immediately after the watchdog timer is expired and no reset has been occurred (Timer mode or Interrupt mode). If the watchdog has been expired in Reset Mode the hardware saves the “watchdog timer expired” status (WTE) over a system restart. That means the WTE must not be cleared by a system reset except a power-on reset. This allows to identify a watchdog caused reset after a system restart and thus distinguish it from other possible reset sources e.g. power up, soft reset, etc. The WTE bit can be cleared via KEAPI call but it cannot be set by any other software operation.

3.19.1 KEApiWatchdogGetCaps

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiWatchdogGetCaps (
      int32_t *pMaxTimeout,
      int32_t *pMinTimeout,
      int32_t *pStagesNr
);
```

Description:

Get the capabilities of the watchdog implementation.

Parameters:

in/out	Parameter name	Description
Out	*pMaxTimeout*	Max. supported watchdog timeout in milliseconds
Out	*pMinTimeout*	Min. supported watchdog timeout in milliseconds
Out	*pStagesNr*	Number of stages the watchdog supports

Returns:

```
KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
```
`KEAPI_RET_ERROR` – other error

3.19.2 KEApiWatchdogSetup

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiWatchdogSetup (
      int32_t stage,
      int32_t timeout,
      int32_t mode
);
```

Description:

Sets up the specified watchdog stage.

Parameters:

in/out	Parameter name	Description
in	*stage*	Stage to be set up (stages are enumerated from 0 to s*tagesNr-1*)
in	*timeout*	Watchdog timeout interval in milliseconds for the specified stage

in	*mode*	The stage mode. This value can either be:
		`KEAPI_WD_MODE_RESET`
		`KEAPI_WD_MODE_INTERRUPT`
		`KEAPI_WD_MODE_TIMER_ONLY`
		`KEAPI_WD_MODE_DISABLE`

Returns:

```
KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
```
`KEAPI_RET_PARAM_ERROR` – *timeout, stage, mode* are out of range, *mode* is unsupported
```
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
```
`KEAPI_RET_ERROR` – other error

3.19.3 KEApiWatchdogEnable

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiWatchdogEnable (void);
```

Description:

Starts the watchdog. All stages should be configured before otherwise undefined behavior.

Parameters:

none

Returns:

```
KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
```
`KEAPI_RET_PARAM_ERROR` – *timeout, delay, mode* are out of range, *mode* is unsupported

KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.19.4 KEApiWatchdogTrigger

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiWatchdogTrigger (void);
```

Description:

Triggers the watchdog timer.

Parameters:

none

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.19.5 KEApiWatchdogDisable

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiWatchdogDisable (void);
```

Description:

Stops the watchdog.

Parameters:

none

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.19.6 KEApiWatchdogGetExpired

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiWatchdogGetExpired (int32_t *pWTE);
```

Description:

Returns Watchdog Timer Expired status (WTE). Returned non-zero signals the watchdog timer has been expired during system runtime (if watchdog is running in no-reset mode) or the system has been restarted after watchdog hardware reset.

Parameters:

in/out	Parameter name	Description
out	*pWTE*	0 – not expired Non-zero: watchdog has been expired

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_PARAM_NULL
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.19.7 KEApiWatchdogClearExpired

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiWatchdogClearExpired (void);
```

Description:

Clears Watchdog Timer Expired status (WTE).

Parameters:

none

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – other error

3.19.8 KEApiWatchdogWaitUntilExpired

```
KEAPI_CALLTYPE KEAPI_RETVAL KEApiWatchdogWaitUntilExpired (void);
```

Description:

This API call shall block execution of current application thread until watchdog interrupt has been occurred. After the watchdog expired and interrupt has been processed by the watchdog driver notification is sent to caller in OS specific way.

Parameters:

none

Returns:

KEAPI_RET_SUCCESS
KEAPI_RET_NOT_INITIALIZED
KEAPI_RET_FUNCTION_NOT_SUPPORTED
KEAPI_RET_FUNCTION_NOT_IMPLEMENTED
KEAPI_RET_ERROR – watchdog has no stage set up in interrupt mode, other error

4. Appendix A: Specification Changes

The list of changes in the specification since last major release.

4.1 Changes from KEAPI release 2.0:

<stdint.h> parameter types instead of KEAPI_INT32, etc.
Use int32_t as type of most parameters, use unsigned types only for binary values and data buffers
Use prefix KEAPI_ or PKEAPI_ for all complex data types (to avoid type conflicts)
No SetCPUPerformance
KEAPI_CPU_FREQUENCY_TURBO is added
No Intel AMT support
No memory modules
No cache, FSB speed and cpuMaxCoreSpeed information in `CPU_INFO`
Remote functionality moved to DMCM
No board handle, no parameters in LibInitialize
Disk partitions are changed to mount points
No SetSystemState
Sensors are redesigned completely, new unified sensors data structures: KEAPI_SENSOR_VALUE and KEAPI_SENSOR_INFO, drop FAN control API
Value of brightness is now in range 0..255
Watchdog API is redesigned completely
GPIO ports introduced which replace direct GPIO pin operations
CPU Frequency and performance information is now returned per CPU core
New `BOARD_INFO` structure
Removed: `I2cWrite`, `I2cRead` (can be implemented via `I2cXfer`)
new API KEApiClearPBITResult and KEApiGetPBITResult introduced
All string fileds and parameters are explicitly specified as zero-terminated strings

5. About Kontron

Kontron is a global leader in embedded computing technology. With more than 30% of its employees in R&D, Kontron creates many of the standards that drive the world's embedded computing platforms. Kontron's product longevity, local engineering, support, and value-added services help to create a sustainable and viable embedded solution for OEMs and system integrators.

Kontron works closely with its customers on their embedded application ready platforms and customer solutions, enabling them to focus on their core competencies. The result is an accelerated time-to-market, reduced total-cost-of-ownership and improved overall application with leading-edge, highly-reliable embedded technology.

Kontron is listed on the German TecDAX stock exchanges under the symbol "KBC".

For more information, please visit: **kontron.com**