K+K

Messtechnik GmbH

Manual KK-Library

Version 19.3.0, 2022-10-09

The KK-Library is available in the following versions:

KK_Library_64.DLL: for Windows 64-bit

KK_FX80E.DLL: for Windows 32-bit

libkk_fx80e.so: for Linux 32-bit

libkk_library_64.so: for Linux 64-bit

Special versions for Python on Linux systems with **cdecl** calling convention:

libkk_library_32_cdecl.so

libkk_library_64_cdecl.so.

This manual describes the multi-source calls introduced with version 18.00, which allow more than one connection to be used at the same time.

Multi-source calls must not be mixed with the previous calls. For new developments, we recommend using only the multi-source calls.

The description of the previous calls can be found in the previous versions of this manual.

Change log

When	Who	Library Version	What
2019-02-06	Loryn	18.00	Initial version of the K+K Multi source library
2019-02-20	Loryn	18.01	Connection type “Simulated data” added
2020-01-23	Loryn	18.01.03	Up to 4 CAN connections per PCAN adapter Enum flag 0x04 for PCAN USB adapter Multi_SetNSZ: value 0 for automatic recognition
2020-03-04	Loryn	18.01.04	Multi_GetReport: error CKK_DLL_CmdIgnored
2020-04-28	Loryn	18.01.07	Data from file: option *WAIT_INTERVAL* added
2020-10-12	Loryn	18.01.10	added: Multi_GetFirmwareVersion, Multi_SetNSZCalibrationData, Multi_HasFRAM, Multi_RemoteLogin NSZ Calibration added TCP-Server added: NSZLOG, NSZDIFFLOG
2020-11-06	Loryn	18.02.02	Multi_SendCommand: reject invalid commands Up to 4 USB connection per USB device (Firmware 63)
2021-03-15	Loryn	18.02.04	TCP-Server added PHASEPREDECESSORLOG, USERLOG1, USERLOG2
2021-05-28	Loryn	19.00.02	Multi_GetReport: error CKK_DLL_Reconnected added: Multi_OpenTcpLogTime
2021-11-18	Loryn	19.01.02	FHR - FXE High Resolution added Multi_IsSerialDevice added Error CKK_DLL_NotSupported if serial connection
2021-12-20	Loryn	19.02.00	added: Multi_OpenTcpLogType, Multi_TcpAppData
2022-04-08	Loryn	19.02.00	Correction C-Syntax Multi_DebugGetFilename
2022-10-09	Loryn	19.03.00	Multi_TcpAppData delivers server response

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

1. Overview

The KK library is primarily used for communication with K+K devices to receive measurement data and messages from and to send control commands to the device.

Furthermore, the data received from the device can be redistributed via server to other applications or retrieved from servers.

The generation of test data and processing of data from files is also supported.

Several parallel connections are possible with KK-Library from version 18.00.

All library calls that use string types (PAnsiChar, AnsiChar) process ASCII-encoded 1-byte characters only.

1.1 Communication with the K+K device

Four different interfaces can be used to communicate with a K+K device

1. Serial via RS232
2. USB connection
3. CAN connection
4. Network connection via TCP/IP

If a connection to a K+K device is to be established, the desired communication path must be selected (see *Connection types*).

1.1.1 Receiving data from the K+K device

For connections via USB and network, a receive thread is started internally in the library, which stores the bytes received by the device in a **receive buffer**. The *Multi_GetReport* call reads from this receive buffer and returns reports (measurement data, messages) in readable form (ASCII format).

With serial connection or CAN connection there is no receiving thread, *Multi_GetReport* reads directly from the device.

In addition to the various interfaces, a K+K device can maintain four connections at the same time. Each connection is assigned a **UserID** (1..4).

1.1.2 Sending control commands to the K+K device

The handling of the commands transferred via *Multi_SendCommand* to the KK-Library depends on the transmission type. Except for the network connection, commands are cached and sent to the K+K device at a given time.

For a network connection, the command is sent directly by *Multi_SendCommand.*

In the case of a serial or CAN connection, the command is sent as soon as a report has been completely received by the device, i.e. the command is sent in the *Multi_GetReport* call.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

For USB (and also for network) connections, the K+K device sends in blocks, which can also contain several reports. For USB connections, the command is sent as soon as a block has been received from the device, i.e. the receive thread sends the commands.

The number of cached commands is initially unlimited. With *Multi_GetPendingCmdsCount* this number can be queried and with *Multi_SetCommandLimit* the number can be limited.

1.2 Local server

For every connection with a K+K device, a local server can be started, which redistributes all data received from the device to the clients logged in to it.

The application may have to be released in the Windows firewall.

1.2.1 Local TCP Server

An application can start a TCP server (*Multi_StartTcpServer*) to provide measurement data strings or log entries via TCP/IP to other applications.

If a TCP server has been started, it is supplied internally with reports (measured data and messages). The TCP server returns reports in readable form (ASCII format) either at the library level (read reports via *Multi_GetReport*) or at LOG level (log entries that the application transfers to the TCP server *via Multi_TcpReportLog*, read via *Multi_GetTcpLog*).

The TCP server distributes in real time the data that is passed to it to the clients that are logged in. If a client is unreachable (connection between server and client has been closed, e.g., client has terminated without logging out), client logon to the TCP server is cleared.

1.3 Connection to a server

In addition to communicating with a K+K device, the KK-Library offers the possibility to connect to a server that has been started by another application with KK-Library. The server can be started on the same computer (Localhost) or on another computer. In this case the KK-Library takes over the role of the client.

The application may have to be released in the Windows firewall.

1.3.1 Connection to a TCP server at the library level

At the library level, the data is read from the TCP server via *Multi_GetReport*. The connection must have been previously opened via *Multi_OpenConnection* (see *Connection Types - Connection to a TCP Server)*.

1.3.2 Connection to a TCP server at the LOG level

An application can start a TCP receiver that receives log entries from a TCP server (*Multi_OpenTcpLog, Multi_OpenTcpLogTime, Multi_OpenTcpLogType*).

When a TCP receiver is started, it internally opens a TCP server socket to receive reports from the TCP server. To do this, he logs on as a client to the TCP server with the desired report mode of the log entries.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

The TCP receiver internally manages a receive buffer of 10,000 entries. If the TCP server sends faster than the application reads (with *Multi_GetTcpLog*), data loss occurs (*Multi_GetTcpLog* reports error *CKK_DLL_BufferOverflow* (8)).

1.3.3 Send AppData to a TCP server

Regardless of whether there is a connection to a TCP server at library level or LOG level, a string can be sent to the application that started the TCP server via *Multi_TcpAppData*. The strings received on the server side are forwarded to the application as a report with header $7F40 the next time GetReport is called.

1.4 Test data

Received reports can be saved as test data in files for later reading and processing.

The test data can be saved as binary reports (**SaveBinaryData*) or as readable ASCII reports (**SaveReportData*). The files are created in the OutputPath directory (see *Multi_SetOutputPath*).

The reading of the test data is implemented as another type of connection. See *Connection types - data from file* to open the connection, read as usual via *Multi_GetReport*.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

1.5 Connection types

A newly opened connection is described by the **OpenString** passed to *Multi_OpenConnection*.

An exception is the connection to a TCP server at the LOG level. There *Multi_OpenTcpLog* is to be used.

In addition to the connection type, a UserID can also be specified.

The OpenString consists of the ConnectionString and a possibly attached UserID. The structure of the ConnectionString is described below.

1.5.1 Local K+K devices connected to the PC by cable

1.5.1.1 Serial interface

ConnectionString = <*SerialName*> with

- <*SerialName*>: Name of the serial interface
  - Windows: COMx, e.g.: 'COM1'
  - Linux: /dev/tty*, e.g.: '/dev/ttyS0'

Note:

To be able to access a serial com interface with Linux, it is necessary for the user to belong to the *dialout* group.

1.5.1.2 USB

ConnectionString = <*DeviceName*> with

- <*DeviceName*> has to match the name delivered by *Multi_EnumerateDevices*

Note:

To be able to access a USB interface with Linux, it is necessary for the user to belong to the *root* group.

(since 18.2)

As of firmware 63, an application can operate up to 4 connections via the same USB interface.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

1.5.1.3 CAN

ConnectionString = CAN:MENLO<Node> with

(since 18.1.3)

ConnectionString = CAN:MENLO<Node>[CHANNEL <Channel>] with

- *<Node>* Device node number (hex 0..F), part of the CAN-ID; CAN-ID is a hard coded setting for Menlo-CAN via PCAN-Basic API
- *<Channel>* Connection number 0..3 for up to 4 different connections. Default number is 0.

Note:

This is possible with Windows and a Peak-Systems PCAN-USB adapter only.

1.5.2 Connection to a K+K device via network

ConnectionString = <IP-Adresse>[:<Port>] with

- *<IP address>* IP address of the K+K device, format of an IPv4 address, e.g. 192.168.178.98
- *[:<Port>]* port number of the local PC to receive measurement data, if not defined the port number is provided by the operating system; port number must be a positive 16 bits value

Note:

The K+K device identifies the connection with the IP address and port number. In order for the K+K device to assign the same transmission buffer when reconnecting, either the UserID must be set or the same port number must be used.

The application may have to be released in the Windows firewall.

1.5.3 Connection to a TCP Server

The K+K device identifies the connection with the IP address and port number. In order for the K+K device to assign the same transmission buffer when reconnecting, either the UserID must be set or the same port number must be used.

ConnectionString = TCP:<IP-Adresse>:<Port> with

- *<IP address>* IP address of the TCP server, format of an IPv4 address, e.g. 192.168.178.98 or 127.0.0.1 for Localhost
- *<Port>* port number of the TCP server, must be a positive 16 bits value

Note:

Receives reports from the TCP server at the library level – measurement data and messages. The TCP server must have been started by an application with KK-Library via *Multi_StartTcpServer*.

The application may have to be released in the Windows firewall.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

1.5.4 Data from a file

ConnectionString = <Filename>[LOOP_FOR_EVER][WAIT_INTERVAL] with

- *<Filename>* File name of a measurement data file
- *[LOOP_FOR_EVER]* optional: data are read in an endless loop always started at the file beginning
- *[WAIT_INTERVAL]* optional: wait time between reports corresponding to report interval

The measurement data file is generated by *Multi_StartSaveBinaryData* for binary data from the K+K device or *Multi_StartSaveReportData* for readable reports of the KK library. The distinction between binary and report data is made via the file name. If the file name ends with KK_ReportData.txt, report data is assumed.

1.5.5 Simulated data

ConnectionString = DemoData

The library generates simulated measurement and FDI data.

1.5.6 UserID

A K+K device can operate up to four connections at the same time. Each connection is assigned a UserID and thus a transmission buffer. The desired UserID can be specified with:

USERID=<no> with

- *<no>* UserID = 1..4

The specification of the UserID is only effective when connecting to a K+K device**, but not with a serial connection**. In order to use the same transmission buffer when reconnecting, the same UserID should be used.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

1.6 Multiple connections

Starting with KK-Library-Version 18.00 the possibility was created to have more than one connection at the same time. For this, the Multi_* routines have been added.

An application that wants to support more than one connection at a time must use these Multi_* calls.

CreateMultiSource must be called per connection. This creates a new administration object internally in the library. *CreateMultiSource* returns a SourceID that must be specified on all subsequent calls and identifies this multi-connection. The administration objects are automatically released when the application ends.

One connection per source is allowed; the connection can be changed.

The Multi_* calls are based on the previous library functions. However, the call must provide the buffer for PAnsiChar parameters. It is assumed that 1024 bytes of buffer are available.

All Multi_* calls are thread safe.

Note:

Multi_* calls must not be mixed with previous calls. For new developments we recommend to use only the Multi_* calls.

1.7 NSZ data and their calibration

If NSZ measurement cards are installed in a K+K device, (D)NSZ data are sent in reports with header $7900. The measured values are reported in nanoseconds per channel (floating point string with two decimal places).
Calibration values are stored in the K+K device's flash memory or F-RAM.

Starting with firmware version 62 these calibration values can be requested with command $02. The K+K device replies with report $7901, which contains a calibration value in nanoseconds per channel (floating point string with three decimal places).

Using the Multi_SetNSZCalibrationData function, NSZ calibration values can be transferred to the K+K device. Requirements are

- Firmware version since 62
- K+K Library version since 18.01.10

If calibration values are changed, the K+K device replies to all users with report $7901.

The TCP server at library level, saves the current $7901 report and sends it to all clients that log on again at library level.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

1.8 FHR – FXE High Resolution

FHR is a multichannel 'dual mixer' type front end for the FXE phase counter, mixing each of its RF inputs with a common local oscillator (LO) to achieve intermediate frequencies (IF) of appr. 500kHz. For each channel, the 20th subharmonic of a low noise 10MHz oscillator is phase locked to its IF, while the phase of those 10MHz is measured by FXE, thereby enhancing the phase resolution by a factor of RF / (RF-LO).

To use FHR cards in a K + K device, the FHR settings per channel are saved in the flash memory or F-RAM of the K + K device.

FHR settings per channel: <Nominal frequency>, <LO frequency>, <enabled>

Nominal and LO frequency are in Hz, enabled: 0 = without FHR, 1 = with FHR

From this, the FHR factor can be calculated for each channel with enabled = 1:

FHR factor = Nominal frequency / (Nominal frequency – LO frequenc<)

Since firmware version 67 FHR settings can be read and written:

- *Multi_ReadFHRData* requests FHR settings; K+K device responds with $7902 report
- *Multi_SetFHRData* writes FHR settings into K+K device.

If FHR settings are changed, the K + K device replies to all users with report $7902.

The TCP server at library level saves the current $7902 report and sends it to all clients that log on at library level for the first time.

$7902 report format:

7902<channel1>/<channel2>/ … /<channel24>

<per channel>:

;<Nominal frequency>;<LO frequency>;<enabled>

<Nominal frequency>, <LO frequency>:

Floating point string in Hz

<enabled>:

0 = without FHR, 1 = with FHR

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

2. Exported functions

All versions of the KK library (KK_FX80E.DLL for Windows 32-bit, KK_Library_64.dll for Windows 64-bit, libkk_fx80e.so for Linux 32-bit, libkk_library_64.so for Linux 64-bit) export the same functions.

All functions have the calling convention **stdcall**:

- Parameter transfer from right to left
- Return values in registers
- Called function clears the stack. Therefore, no variable argument lists are possible.
- All exported routines are described here in Pascal notation. Chapter 4 describes the function declarations in C.

Note:

Python on Linux systems needs **cdecl** calling conventions. This is implemented in KK libraries libkk_library_32_cdecl.so and libkk_library_64_cdecl.so.

2.1 Create multiple connection

- function **CreateMultiSource**: Integer; stdcall;

2.2 List available interfaces

- function **Multi_EnumerateDevices**(Names: PAnsiChar; EnumFlags: Byte): Integer; stdcall;
- function **Multi_GetEnumerateDevicesErrorMsg**: PAnsiChar; stdcall;
- function **Multi_GetHostAndIPs**(HostName: PAnsiChar; IPaddr: PAnsiChar; ErrorMsg: PAnsiChar): Integer; stdcall;

2.3 Path description

- function **Multi_GetOutputPath**(ID: Integer): PAnsiChar; stdcall;
- function **Multi_SetOutputPath**(ID: Integer; Path: PAnsiChar): PAnsiChar; stdcall;

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

2.4 Debug protocol

- function **Multi_Debug**(ID: Integer; DbgOn: Boolean; DbgID: PAnsiChar): PAnsiChar; stdcall;
- function **Multi_DebugFlags**(ID: Integer; ReportLog: Boolean; LowLevelLog: Boolean): Integer; stdcall;
- function **Multi_DebugLogLimit**(ID: Integer; LogType: Byte; aSize: Cardinal): Integer; stdcall;
- function **Multi_DebugGetFilename**(ID: Integer): PAnsiChar; stdcall;

2.5 Info queries

- function **Multi_GetDLLVersion**: PAnsiChar; stdcall;
- function **Multi_GetBufferAmount**(ID: Integer): Integer; stdcall;
- function **Multi_GetTransmitBufferAmount**(ID: Integer): Integer; stdcall;
- function **Multi_GetUserID**(ID: Integer): Byte; stdcall;
- function **Multi_IsFileDevice**(ID: Integer): Boolean; stdcall;
- function **Multi_IsSerialDevice**(ID: Integer): Boolean; stdcall;
- function **Multi_GetFirmwareVersion**(ID: Integer): Integer; stdcall;
- function **Multi_HasFRAM**(ID: Integer): Boolean; stdcall;

2.6 Open and close a connection

- function **Multi_OpenConnection**(ID: Integer; Connection: PAnsiChar; BlockingIO: Boolean): Integer; stdcall;
- procedure **Multi_CloseConnection**(ID: Integer); stdcall;

2.7 Read reports

- function **Multi_SetDecimalSeparator**(ID: Integer; Separator: AnsiChar): Integer; stdcall;
- function **Multi_SetNSZ**(ID: Integer; aNSZ: Integer): Integer; stdcall;
- function **Multi_GetReport**(ID: Integer; Data: PAnsiChar): Integer; stdcall;

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

2.8 Send commands

- function **Multi_GetPendingCmdsCount**(ID: Integer): Cardinal; stdcall;
- function **Multi_SetCommandLimit**(ID: Integer; Limit: Cardinal): Integer; stdcall;
- function **Multi_SendCommand**(ID: Integer; Command: PAnsiChar; Len: Integer): Integer; stdcall;
- function **Multi_RemoteLogin**(ID: Integer; Password: LongWord; err: PAnsiChar): Integer; stdcall;

2.9 Local TCP server

- function **Multi_StartTcpServer**(ID: Integer; **var** aPort: Word): Integer; stdcall;
- function **Multi_StopTcpServer**(ID: Integer): Integer; stdcall;
- function **Multi_GetTcpServerError**(ID: Integer): PAnsiChar; stdcall;
- procedure **Multi_TcpReportLog**(ID: Integer; Data: PAnsiChar; logType: Integer); stdcall;

2.10 Connection to a TCP server on LOG level

- function **Multi_OpenTcpLog**(ID: Integer; IpPort: PAnsiChar; Mode: PAnsiChar): Integer; stdcall;
- function **Multi_OpenTcpLogTime**(ID: Integer; IpPort: PAnsiChar; Mode: PAnsiChar; Format: PAnsiChar): Integer; stdcall;
- function **Multi_OpenTcpLogType**(ID: Integer; IpPort: PAnsiChar; LogType: Integer; Format: PAnsiChar): Integer; stdcall;
- procedure **Multi_CloseTcpLog**(ID: Integer); stdcall;
- function **Multi_GetTcpLog**(ID: Integer; Data: PAnsiChar): Integer; stdcall;

2.11 Send AppData to a TCP server

- function **Multi_TcpAppData**(ID: Integer; Data: PAnsiChar): Integer; stdcall;

2.12 Generate test data

- function **Multi_StartSaveBinaryData**(ID: Integer; DbgID: PAnsiChar): Integer; stdcall;
- function **Multi_StopSaveBinaryData**(ID: Integer): Integer; stdcall;
- function **Multi_StartSaveReportData**(ID: Integer; DbgID: PAnsiChar): Integer; stdcall;
- function **Multi_StopSaveReportData**(ID: Integer): Integer; stdcall;

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

2.13 NSZ Calibration

- function **Multi_SetNSZCalibrationData**(ID: Integer; Data: PAnsiChar): Integer; stdcall;

2.14 FHR-Einstellungen

- function **Multi_ReadFHRData**(ID: Integer): Integer;
- function **Multi_SetFHRData**(ID: Integer; Data: PAnsiChar): Integer;

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3. Functional description

Function names, invocation parameters and return values are identical in all versions (*KK_FX80E.DLL* for Windows 32-bit, *KK_Library_64.dll* for Windows 64-bit, *libkk_fx80e.so* for Linux).

3.1 Create multiple connection

3.1.1 CreateMultiSource

function CreateMultiSource: Integer; stdcall;

Creates a new management object and returns a SourceID. This SourceID must be specified on subsequent Multi_* calls.

Returns:

- *SourceID* (> = 0)

Note:

All administration objects are automatically released when the program ends.

3.2 List available interfaces

3.2.1 Multi_EnumerateDevices

function Multi_EnumerateDevices (Names: PAnsiChar; EnumFlags: Bytes): Integer; stdcall;

Lists the interfaces / K+K devices selected via *EnumFlags* and returns them in *Names*.

Parameters:

- *Names*: contains found entries separated by comma or 0 if no interface / device was found. Must point to a 1024 byte buffer, output in *Names* may be truncated to 1024 bytes
- *EnumFlags*: defines elements to search, see table below

Returns:

- *0:* no error
- *<0:* Error flag similar to EnumFlags: An error occurred while listing the corresponding interfaces / devices.

If an error occurs, use *Muti_GetEnumerateDevicesErrorMsg* to get the error message.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

EnumFlags

Name	Value	Description
EnumFlag_ComPorts	$01	List all serial ports (includes virtual ports via USB adapters)
EnumFlag_USB	$02	List all K+K USB devices (HID + CDC)
EnumFlag_LocalDevices	$03	List serial ports and K+K USB devices
EnumFlag_PCAN	$04	PCAN USB Adapter
EnumFlag_All	$FF	List all

3.2.2 Multi_GetEnumerateDevicesErrorMsg

function Multi_GetEnumerateDevicesErrorMsg: PAnsiChar; stdcall;

Returns the error message or null pointer generated by *Multi_EnumerateDevices*.

Returns:

- *nil:* no error message
- *<>nil:* error message

3.2.3 Multi_GetHostAndIPs

function Multi_GetHostAndIPs (HostName: PAnsiChar; IPaddr: PAnsiChar; ErrorMsg: PAnsiChar): Integer; stdcall;

Specifies your own host name and all IP addresses (separated by commas).

Unlike the other Multi_* calls, 80 bytes of buffer size are sufficient here

Parameters:

- *HostName:* must point to a buffer with 80 bytes, contains its own hostname after call or 0 in case of error
- *IPaddr:* must point to a buffer with 80 bytes, contains after calling its own IP addresses separated by a comma or 0 in case of error
- *ErrorMsg:* must point to a buffer with 80 bytes, contains 0 or in case of error an error message

Returns:

- *CKK_DLL_NoError (1):* ok
- *CKK_DLL_BufferTooSmall (6):* One or more return strings truncated to 80 characters
- *CKK_DLL_Error (0):* Error, ErrorMsg contains an error message

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.3 Paths

3.3.1 Multi_GetOutputPath

function Multi_GetOutputPath (ID: Integer): PAnsiChar; stdcall;

Returns current file output path (debug log, test data) for source *ID*.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

Returns:

- currently set output path for source *ID*
- Error message "Source ID <ID> not found"

3.3.2 Multi_SetOutputPath

function Multi_SetOutputPath (ID: Integer; Path: PAnsiChar): PAnsiChar; stdcall;

Sets file output path (debug log, test data) for source *ID* and tests access rights. Default is the path of the EXE file.

The output path ONLY affects files to be created in the future; already open files retain their file path.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *Path*: output path, path separator is appended if necessary

Returns:

- *nil:* ok
- <> *nil:* Error message, output path was not set

Note:

To create a unique file name, either different output paths must be set per SourceID or different DbgIDs must be specified when creating the files (see *Multi_Debug, Multi_StartSave* * Data).

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.4 Debug log

A library-internal debug log can be created that can be opened, closed, and configured with the following functions. One debug protocol exists per source, i.e. different management objects write to separate files.

3.4.1 Multi_Debug

function Multi_Debug (ID: Integer; DbgOn: Boolean; DbgID: PAnsiChar): PAnsiChar; stdcall;

Opens / closes library-internal debug log for source *ID*. By default, the debug log is written to a file *KK_DLL_LOG_ <DbgID> .txt* (the same directory as the application or to the directory specified by *Multi_SetOuputPath*). Any existing file will be overwritten. See *Multi_DebugLogLimit* for advanced settings of the debug log. If the debug file could not be created (for example, missing access rights), an error message is returned.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *DbgOn: False* closes the log, *True* opens it
- *DbgID*: Source ID of the debug log, part of the file name

Returns:

- *nil:* ok
- <> *nil:* error message

Note:

DbgID is used to differentiate the debug logs to different sources to generate unique file names. This is especially necessary if the output directories are the same. If not working with multiple sources*, DbgID* can also be nil.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.4.2 Multi_DebugFlags

function Multi_DebugFlags (ID: Integer; ReportLog: Boolean; LowLevelLog: Boolean): Integer; stdcall;

Modifies scope of debug log for source *ID*, Default: ReportLog = True, LowLevelLog = False. The new settings take effect immediately and can be changed as required during operation.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *ReportLog* = *True*: reports supplied by *Multi_GetReport* are written to the debug log
- *LowLevelLog* = *True*: Data stream sent and received over the connection is written to the debug log.

Returns:

- *CKK_DLL_NoError* (1): ok
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID

3.4.3 Multi_DebugLogLimit

function Multi_DebugLogLimit (ID: Integer; LogType: Byte; aSize: Cardinal): Integer; stdcall;

Sets limits for debug log of source *ID*, default: LogType = 0, aSize = 0. For these settings to take effect, they must be set before the debug log is opened with *Multi_Debug.*

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *LogType*: see table below
- *aSize*: max. Size of the log in number of bytes, aSize = 0: Size is set to 20 MB

Returns:

- CKK_DLL_NoError (1): ok
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

Values for LogType

Name	Value	Description
logUnlimited	0	A debug protocol is generated. The file is unlimited in size (default). The value for aSize is ignored. File name: *KK_DLL_LOG_ <DbgID> .txt*
logOverwrite	1	The size of the debug log is limited (aSize = 0: default 20MB). When the size is reached, the log is overwritten from the beginning. File name: *KK_DLL_LOG_ <DbgID> .txt*
logCreateNew	2	The debug log is limited in size (aSize = 0: default 20MB). When the size is reached, a new file is created. File name contains date and time of creation (*KK_DLL_LOG_ <DbgID> _ <yyyymmdd> _ <hhmmss> .txt*).

3.4.4 Multi_DebugGetFilename

function Multi_DebugGetFilename (ID: Integer): PAnsiChar; stdcall;

Returns the absolute file name of the current source ID debug log file.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

Returns:

- current file name; *nil* if no debug log is written
- Error message "Source ID <ID> not found"

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.5 Info queries

3.5.1 Multi_GetDLLVersion

function Multi_GetDLLVersion: PAnsiChar; stdcall;

Returns:

- KK library date and version.

3.5.2 Multi_GetBufferAmount

function Multi_GetBufferAmount (ID: Integer): Integer; stdcall;

Returns the fill level of the receive buffer for source *ID* in number of bytes

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

Returns:

- *0:* buffer is empty or error (*ID* invalid, no active connection)
- *>0:* Number of unread bytes in the receive buffer for source *ID*

3.5.3 Multi_GetTransmitBufferAmount

function Multi_GetTransmitBufferAmount (ID: Integer): Integer; stdcall;

Returns the fill level of the transmit buffer in the K+K device connected to source *ID*.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

Returns:

- *0:* buffer is empty or error (ID invalid, no connection to a K+K device)
- *>0:* Number of bytes not yet sent in the transmission buffer of the K+K device connected to source *ID*

Note:

For a serial connection, the fill level of the transmission buffer can't be determined; 0 is always delivered.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.5.4 Multi_GetUserID

function Multi_GetUserID (ID: Integer): Byte; stdcall;

Returns the UserID assigned by the K+K device. See *Connection Types*.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

Returns:

- *1..4:* UserID assigned by the K+K device
- *255:* Error (*ID* invalid, no connection to a K+K device)

3.5.5 Multi_IsFileDevice

function Multi_IsFileDevice (ID: Integer): Boolean; stdcall;

When processing data from a file (see *Connection Types - Data from File*) *True* is returned here. If *Multi_GetReport* reports the error *CKK_DLL_DeviceNotConnected* (7), the read process should be terminated because no more data is being delivered.

Parameters:

- *ID*: SorceID to which this call relates, value supplied by CreateMultiSource

Returns:

- Measurement data is read from a file

3.5.6 Multi_IsSerialDevice

function Multi_IsSerialDevice (ID: Integer): Boolean; stdcall;

For serial connections *True* is returned here. This can be used to check whether CKK_DLL_NotSupported for Multi_SetNSZCalibrationData, Multi_SetFHRData denotes a not supported firmware version or a serial connection.

Parameters:

- *ID*: SorceID to which this call relates, value supplied by CreateMultiSource

Returns:

- serial connection

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.5.7 Multi_GetFirmwareVersion

function Multi_GetFirmwareVersion(ID: Integer): Integer; stdcall;

(since 18.01.10)

Delivers the firmware versions number from a version report received previously. A versions report (header $7001) is requested with command $01 or $81.

Parameters:

- *ID:* Sorce-ID supplied by *CreateMultiSource*

Returns:

- *-1:* error: Source-ID invalid or no versions report has yet been received
- *> 1:* firmware version number

3.5.8 Multi_HasFRAM

function Multi_IsFileDevice(ID: Integer): Boolean; stdcall;

(since 18.01.10)

Returns K+K device is equipped with an F-RAM ausgestattet (see *NSZ Calibration*). A previously received versions report is required for this. A versions report (header $7001) is requested with command $01 or $81.

Parameters:

- *ID:* Sorce-ID supplied by *CreateMultiSource*

Liefert:

- *False:* K+K device has no F-RAM or there is no versions report
- *True:* K+K device has F-RAM (e.g. for NSZ calibration data)

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.6 Open connection, close

3.6.1 Multi_OpenConnection

function Multi_OpenConnection (ID: Integer; Connection: PAnsiChar; BlockingIO: Boolean): Integer; stdcall;

Opens the connection described in *Connection* for source *ID*. A previous connection for source *ID* is closed.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *Connection*: see connection types, must point to a buffer with 1024 bytes, contains an error message in case of error
- *BlockingIO* = *True*: opens connection with blocking read / write

Returns:

- *CKK_DLL_NoError* (1): ok
- Otherwise error number and connection contains an error message:
- *CKK_DLL_Error* (0): error
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID

Note Blocking IO:

BlockingIO leads to blocking I / O operations. This means that all read / write calls return after a waiting time (timeout: approx. 200 ms) if nothing has been received. *BlockingIO* is NOT supported by data from file.

It should work with its own read thread (to do not block the application).

Note *(since 18.2)*:

As of firmware 63, an application can operate up to 4 connections via the same USB interface.

To be considered under Linux:

In order to access a locally connected device under Linux, certain access rights are required:

- USB: the user must belong to the root group
- Serial: the user must belong to the group *dialout*
- Linux command (this requires root privileges!) to add a user to a group: usermod -a -G <group> <user>

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.6.2 Multi_CloseConnection

procedure Multi_CloseConnection (ID: integer); stdcall;

Closes current connection for source *ID*. Any commands not yet sent will be deleted.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

3.7 Reading in reports

3.7.1 Multi_SetDecimalSeparator

function Multi_SetDecimalSeparator (ID: Integer; Separator: AnsiChar): Integer; stdcall;

The KK library returns measured values (floating point numbers) received by the K+K device as an AnsiString. Here, the decimal separator for source ID to be used in the representation of a floating-point number can be specified. This decimal separator is also used to convert strings to floating point values.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

Separator: decimal separator, values other than '.' or ',' are ignored

Returns:

- *CKK_DLL_NoError* (1): ok
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID

3.7.2 Multi_SetNSZ

function Multi_SetNSZ (ID: Integer; aNSZ: Integer): Integer; stdcall;

Depending on the measurement card used, NSZ reports are transmitted as single NSZ or double NSZ. This function defines for source *ID* whether a single or double NSZ is used.

(since 18.1.3)

Value 0 for aNSZ: automatic recognition of the NSZ format

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *aNSZ*: 1 = Single-NSZ, 2 = Double-NSZ, other values are ignored

Returns:

- *CKK_DLL_NoError* (1): ok
- CKK_*DLL_SourceNotFound* (10): *ID* is not a valid source ID

Note:

Calls described in former manual:

- *FX_GetReport* expects single NSZ
- *LE_ReadMonPha* expects Double NSZ

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.7.3 Multi_GetReport

function Multi_GetReport (ID: Integer; Data: PAnsiChar): Integer; stdcall;

Returns the next received report in Data for source *ID*.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *Data*: must point to a buffer with 1024 bytes, contains the next report (measurement data, message) or 0 after calling, if no report exists or an error message in the event of an error

Returns:

- *CKK_DLL_NoError* (1): ok
- *CKK_DLL_BufferTooSmall* (6): Report in Data was shortened to 1024 bytes
- Otherwise error number and data contains an error message:
- *CKK_DLL_Error* (0): error
- *CKK_DLL_WriteError* (3): Error during command transmission, connection has been closed. see *Multi_SendCommand*
- *CKK_DLL_ServerDownError* (4): Destination server not reachable, connection was closed
- *CKK_DLL_DeviceNotConnected* (7): no connection available
- *CKK_DLL_HardwareFault* (9): Peer does not send measurement data, connection was closed
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID
- *CKK_DLL_CmdIgnored(11)*: Commad wasn't sent (simulator refuses command)
- *CKK_DLL_Reconnected*(13): Reconnection of an interrupted connection with data loss

Note:

For errors 3 and 4, the application should reopen the connection (call M*ulti_OpenConnection* again).

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.8 Send command

3.8.1 Multi_GetPendingCmdsCount

function Multi_GetPendingCmdsCount (ID: Integer): Cardinal; stdcall;

Returns the number of cached commands for source *ID*

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

Returns:

- *0:* no waiting commands or errors (ID invalid, no active connection)
- *>0:* Number of cached commands

3.8.2 Multi_SetCommandLimit

function Multi_SetCommandLimit (ID: Integer; Limit: Cardinal): Integer; stdcall;

Sets the length of the command queue for source *ID* to *Limit*. With *Limit* = *0*, the number of commands is unlimited, default = 0

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *Limit*: number of commands in the queue (0 = unlimited)

Returns:

- *CKK_DLL_NoError* (1): ok
- C*KK_DLL_SourceNotFound* (10): *ID* is not a valid source ID

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.8.3 Multi_SendCommand

function Multi_SendCommand (ID: Integer; Command: PAnsiChar; Len: Integer): Integer; stdcall;

Inserts *Command* into the source *ID* command queue. If there is no active connection or the command list is full (*Multi_SetCommandLimit*), the command is rejected (*CKK_CmdIgnored*).

The command is sent on the next *Multi_GetReport* call. In the event of an error, the corresponding error message is returned.

For a network connection, the command is sent immediately. In the event of an error, the corresponding error message is already returned here.

(since 18.2):

Invalid commands are rejected with error *CKK_CmdIgnored (11)*.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *Command:* command to send, in case of error Command contains an error message, must point to a buffer with 1024 bytes
- *Len*: Command length in bytes

Returns*:*

CKK_DLL_NoError (1): ok

otherwise error number and Command contains an error message:

- *CKK_DLL_Error* (0): error
- *CKK_DLL_SourceNotFound* (10): ID is not a valid source ID
- *CKK_CmdIgnored* (11): Command was rejected

Note:

Commands are not sent for data from file and when connected to a TCP server.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.8.4 Multi_RemoteLogin

function Multi_RemoteLogin(ID: Integer; Password: LongWord; err: PAnsiChar): Integer; stdcall;

If there is a network connection to the K+K device, *Password* is encrypted and sent in a remote login command (see *Multi_SendCommand*), otherwise the function is terminated without errors.

Parameters:

- *ID: SourceID supplied by CreateMultiSource*
- *Password:* unencrypted password, integer value in range 0..4294967295
- *err:* must point to a buffer with 1024 bytes, contains an error message if function fails

Returns:

- *CKK_DLL_NoError (1):* ok

  otherwise error number and *err* contains an error message:
- *CKK_DLL_Error(0):* login denied (wrong password)
- *CKK_DLL_SourceNotFound(10): ID* is no a valid source ID
- *CKK_CmdIgnored(11): c*ommand was rejected

Note:

Remote login is neccessary, if flash memory or F-RAM of the K+K device is to be changed via a network connection.

Example: *Multi_SetNSZCalibrationData*

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.9 Local TCP Server

3.9.1 Multi_StartTcpServer

function Multi_StartTcpServer (ID: Integer; var aPort: Word): Integer; stdcall;

Starts local TCP server for source *ID* on TCP port *aPort*. In case of error, an error message can be retrieved via *Multi_GetTcpServerError*.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *aPort*: TCP port number on which the server can be reached, 0: System assigns port number, this is returned

Returns:

- *CKK_DLL_NoError* (1): ok
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID
- *CKK_DLL_Error* (0): Error, retrieve error message via *Multi_GetTcpServerError*

Note:

The TCP server can only be reached via its server port number. It must be made public to enable the clients to connect to the TCP server.

3.9.2 Multi_StopTcpServer

function Multi_StopTcpServer (ID: Integer): Integer; stdcall;

Stops local TCP server for source *ID*. All existing client connections will be terminated. If an error occurs, an error message can be retrieved via *Multi_GetTcpServerError*.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

Returns:

- *CKK_DLL_NoError* (1): ok
- *CKK_DLL_SourceNotFound* (10): ID is not a valid source ID
- *CKK_DLL_Error* (0): Error, retrieve error message via *Multi_GetTcpServerError*

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.9.3 Multi_GetTcpServerError

function Multi_GetTcpServerError (ID: Integer): PAnsiChar; stdcall;

Returns an error message stored in the TCP server.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

Returns:

- *nil*: There is no error message
- <> *nil*: error message

3.9.4 Multi_TcpReportLog

procedure Multi_TcpReportLog (ID: Integer; Data: PAnsiChar; logType: Integer); stdcall;

Submit log entry to local TCP server for redistribution. If *ID* is not a valid source ID or there is no TCP server for this ID or *Data* = nil, the call is ignored.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *Data*: log entry to be redistributed
- *logType*: classifies log entry for forwarding to TCP clients. When logging in to the TCP server, the clients specify a report mode that specifies which log entries are sent to the client. *logType* and report mode are as follows:

logType	Report mode
0	PHASELOG
1	FREQLOG
2	PHASEDIFFLOG
3	NSZLOG
4	NSZDIFFLOG
5	PHASEPREDECESSORLOG
6	USERLOG1
7	USERLOG2

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.10 Connection to a TCP server at LOG level

3.10.1 Multi_OpenTcpLog

function Multi_OpenTcpLog (ID: Integer; IpPort: PAnsiChar; Mode: PAnsiChar): Integer; stdcall;

Generates a TCP receiver for receiving log entries in report mode *Mode* from a TCP server that is addressed with *IpPort.* There is a client login to the TCP server.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *IpPort*: IP address and port number of the TCP server in the format <Ipv4>: <port>; in the event of an error, IpPort contains an error message that must point to a buffer with 1024 bytes
- *Mode*: report mode to report, one of the strings
  - PHASELOG
  - FREQLOG
  - PHASEDIFFLOG
  - NSZLOG
  - NSZDIFFLOG
  - PHASEPREDECESSORLOG
  - USERLOG1
  - USERLOG2

Returns

- *CKK_DLL_NoError* (1): ok
- otherwise error number and *IpPort* contains an error message
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID
- *CKK_DLL_Error* (0): error (for example, report mode not available)

Note:

If the TCP server can no longer reach the client (connection between client and server has been closed), the client login in the TCP server is deleted.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.10.2 Multi_CloseTcpLog

procedure Multi_CloseTcpLog (ID: integer); stdcall;

Ends TCP receiver for source *ID*. TCP receiver logs off the TCP server.

After closing, it is still possible to retrieve existing data (*Multi_GetTcpLog*).

If *ID* is not a valid source ID or there is no TCP receiver for this ID, the call is ignored.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

3.10.3 Multi_GetTcpLog

function Multi_GetTcpLog (ID: Integer; Data: PAnsiChar): Integer; stdcall;

Returns the next log entry received by the TCP receiver in *Data*.

The TCP receiver internally contains a receive buffer of 10,000 entries. If the TCP server sends faster than the application reads (with *Multi_GetTcpLog*), data loss occurs. The TCP receiver discards received data when the receive buffer is full.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *Data*: must point to a buffer with 1024 bytes, contains after call:
  - 0: no report available
  - Report
  - Error message in case of error

Returns:

- *CKK_DLL_NoError* (1): ok
- *CKK_DLL_BufferTooSmall* (6): Log entry in Data has been truncated to 1024 bytes
- Otherwise error number and data contains an error message:
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID
- *CKK_DLL_ServerDownError* (4): TCP server has closed connection
- *CKK_DLL_BufferOverflow* (8): There was a data loss

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.10.4 Multi_OpenTcpLogTime

function Multi_OpenTcpLogTime(ID: Integer; IpPort: PAnsiChar; Mode: PAnsiChar; Format: PAnsiChar): Integer; stdcall;

(since 19.00.02)

Like *Multi_OpenTcpLog* with the difference that the received log entries have a prepended UTC time stamp, which is formatted with the format string specified in the *Format* parameter.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *IpPort*: IP address and port number of the TCP server in the format <Ipv4>: <port>; in the event of an error, IpPort contains an error message that must point to a buffer with 1024 bytes
- *Mode*: report mode to report, see *Multi_OpenTcpLog*
- *Format:* specifies format of UTC timestamp

Returns

- *CKK_DLL_NoError* (1): ok
- otherwise error number and *IpPort* contains an error message
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID
- *CKK_DLL_Error* (0): error (for example, report mode not available)

Note:

If the TCP server can no longer reach the client (connection between client and server has been closed), the client login in the TCP server is deleted.

Example format:

YYYYMMDD HH:NN:SS.ZZZ with

- YYYY: Year 4-digit
- MM: Month
- DD: Day
- HH: Hour (24 hour indication)
- NN: Minutes
- SS: Seconds
- ZZZ: Milliseconds

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.10.5 Multi_OpenTcpLogType

function Multi_OpenTcpLogType(ID: Integer; IpPort: PAnsiChar; LogType: Integer; Format: PAnsiChar): Integer; stdcall;

(since 19.02.00)

Analogous to *Multi_OpenTcpLog*, *Multi_OpenTcpLogTime* with the difference that the desired mode is specified as an integer (0..7).

If the format is not equal to nil, the received log entries are preceded by a time stamp in UTC.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *IpPort*: IP address and port number of the TCP server in the format <Ipv4>: <port>; in the event of an error, IpPort contains an error message that must point to a buffer with 1024 bytes
- *LogType*: report mode to be registered (0..7)
- *Format:* specifies format of UTC timestamp (nil: without timestamp)

Returns

- *CKK_DLL_NoError* (1): ok
- otherwise error number and *IpPort* contains an error message
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID
- *CKK_DLL_Error* (0): error (for example, report mode not available)

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.11 Send AppData to a TCP server

function Multi_TcpAppData(ID: Integer; Data: PAnsiChar): Integer; stdcall;

(since 19.02.00)

(since 19.03.00 with server response)

Sends the string contained in Data to the TCP server connected for Source ID at library level or log level. The receiving TCP server forwards the data to the application as a report with header $7F40 the next time GetReport is called.

As of version 19.3.0, Data contains the server response after the call.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *Data:* contains the string to be transmitted, must point to a buffer with 1024 bytes. Contains server response after call or error message in case of error.

Returns:

- *CKK_DLL_NoError* (1): ok, *Data* contains server response
- otherwise error number and *Data* contains an error message
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID
- *CKK_DLL_ServerDownError (4): No connection to* TCP server
- *CKK_DLL_Error(0):* error reported by TCP server

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.12 Generate test data

Received reports can be saved as test data in files for a later processing.

The test data can be saved as binary reports (** SaveBinaryData*) or as readable ASCII reports *(* SaveReportData*). The files are created in the directory *OutputPath* (see *Multi_SetOutputPath*).

The reading of the test data is implemented as another type of connection. See *Connection types - Data from file* to open the connection, read as usual *via Multi_GetReport*.

3.12.1 Multi_StartSaveBinaryData

function Multi_StartSaveBinaryData (ID: Integer; DbgID: PAnsiChar): Integer; stdcall;

Starts the logging of received reports for source *ID* into binary file *<Date>_<Time>_<DbgID>_KK_BinaryData.bin* in the *OutputPath* directory (see *Multi_SetOutputPath*). In case of an error, nothing is stored.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *DbgID*: Source identifier of the binary data, part of the file name

Returns:

- *CKK_DLL_NoError* (1): ok
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID
- *CKK_DLL_DeviceNotConnected* (7): no active connection

Note:

DbgID is used to differentiate the binary data to different sources to generate unique file names. This is necessary if the output directories are the same. If not working with multiple sources, *DbgID* can also be nil.

3.12.2 Multi_StopSaveBinaryData

function Multi_StopSaveBinaryData (ID: Integer): Integer; stdcall;

Stops saving binary reports for source *ID* and closes the binary file.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

Returns:

- *CKK_DLL_NoError* (1): ok
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID
- *CKK_DLL_DeviceNotConnected* (7): no active connection

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.12.3 Multi_StartSaveReportData

function Multi_StartSaveReportData (ID: Integer; DbgID: PAnsiChar): Integer; stdcall;

Starts the logging of reported reports for source ID into text file *<Date>_<Time>_<ID>_KK_ReportData.txt* in directory *OutputPath* (see *Multi_SetOutputPath*).

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*
- *DbgID*: Source ID of the report data, part of the file name

Returns:

- *CKK_DLL_NoError* (1): ok
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID

Note:

DbgID is used to differentiate the report data to different sources to generate unique file names. This is especially necessary if the output directories are the same. If not working with multiple sources, *DbgID* can also be nil.

3.12.4 Multi_StopSaveReportData

function Multi_StopSaveReportData (ID: Integer): Integer; stdcall;

Stops saving the reports for source *ID* and closes the text file.

Parameters:

- *ID*: SourceID supplied by *CreateMultiSource*

Returns:

- *CKK_DLL_NoError* (1): ok
- *CKK_DLL_Error* (0): file was not open
- *CKK_DLL_SourceNotFound* (10): *ID* is not a valid source ID

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.13 NSZ Calibration

From firmware version 62, calibration data can be saved, queried and changed in the K+K device. Changing the calibration data via a network connection is only possible after prior remote login. In the case of a serial connection, the change is rejected.

Depending on the hardware equipment, the calibration data are written to the flash memory or to an additional F-RAM. The presence of an F-RAM is indicated in the version report by the identifier "F", see also *Multi_HasFRAM*.

Changes to the flash memory always lead to a disturbance of the FXE measured values and, depending on the timing, also to a disturbance of the NSZ measured values.

It is recommended to check the version and equipment of the K+K device before changing the calibration data: Command $01 or $81 requests the version report $7001.

3.13.1 Multi_SetNSZCalibrationData

function Multi_SetNSZCalibrationData(ID: Integer; Data: PAnsiChar): Integer; stdcall;

(since 18.01.10)

Transmits NSZ calibration data via source connection *ID* (not supported. Serial connection) to the K+K device, where they are stored in flash or F-RAM. The K+K device informs all users about changed calibration data by means of a $7901 report.

Data contains the calibration values per channel (maximum 24 channels) separated by semicolons. The calibration values must be specified in nanosecunds (floating point string with a maximum of 3 decimal places).

Parameters:

- *ID: SourceID supplied by CreateMultiSource*
- *Data:* floating point string with a maximum of 3 decimal places in nanosecunds per channel, separated by semicolons

Returns:

- *CKK_DLL_NoError(1):* ok
- *CKK_DLL_Error(0):* format error *Data*: more than 24 channels or converting into float fails, see *Multi_SetDecimalSeparator*
- *CKK_DLL_SourceNotFound(10): ID* is not a valid source ID
- *CKK_DLL_NotSupported(12):* firmware version < 62*;* serial connection not supported

Note:

If the *ID* represents a network connection, the caller must have previously authenticated himself through a remote login (see *Multi_RemoteLogin*). Otherwise the change will not be applied and there is no $7901-Report as an answer.

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

Caution:

Changes to the flash memory always lead to a disturbance of FXE measured values and, depending on the timing also to a disturbance of NSZ measured values

3.14 FHR settings

Since firmware version 67, FHR settings can be saved, queried and changed in the K + K device.

Changing the FHR settings via a network connection is only possible after prior remote login. With a serial connection, the FHR settings can be read, but not changed.

Depending on the hardware equipment, the FHR settings are written to the flash memory or to an additional F-RAM. The presence of an F-RAM is indicated in the version report by the identifier "F", see also Multi_HasFRAM.

Changes to the flash memory always lead to a disruption of the FXE measured values and, depending on the timing, also to a disruption of the NSZ measured values.

It is recommended to check the version and equipment of the K + K device before changing the FHR settings: Command $ 01 or $ 81 requests the version report $ 7001.

3.14.1 Multi_ReadFHRData

function Multi_ReadFHRData(ID: Integer): Integer; stdcall;

(since 19.1.2)

Requests FHR settings for source *ID*.

If successful, K+K device reponds with a $7902 report.

Parameter:

- *ID:* SourceID supplied by *CreateMultiSource*

Returns:

- *CKK_DLL_NoError(1):* ok
- *CKK_DLL_Error(0):* command failed
- *CKK_DLL_SourceNotFound(10): ID* is not a valid source ID
- *CKK_DLL_NotSupported(12):* firmware version < 67*;* serial connection not supported

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

3.14.2 Multi_SetFHRData

function Multi_SetFHRData(ID: Integer; Data: PAnsiChar): Integer; stdcall;

(since 19.1.2)

Transmits FHR settings via source connection ID (except serial connection) to the K + K device, where they are saved in flash or F-RAM. The K + K device informs all users about changed FHR settings by means of a $ 7902 report.

Data contains the FHR settings per channel (maximum 24 channels) separated by a slash. Nominal frequency, LO frequency and enabled per channel separated by semicolons. The frequency values must be specified in Hz (floating point string). Either 0 or 1 is enabled.

Parameters:

- *ID:* SourceID supplied by *CreateMultiSource*
- *Data*: Floating point string with FHR settings per channel (maximum 24 channels)

Returns:

- *CKK_DLL_NoError(1):* ok
- *CKK_DLL_Error(0):* Format error
- *CKK_DLL_SourceNotFound(10): ID* is not a valid source ID
- *CKK_DLL_NotSupported(12):* firmware version < 67*;* serial connection not supported

Note:

If the *ID* represents a network connection, the caller must have previously authenticated himself through a remote login (see *Multi_RemoteLogin*). Otherwise the change will not be applied and there is no $7902-Report as an answer.

Caution:

Changes to the flash memory always lead to a disturbance of FXE measured values and, depending on the timing also to a disturbance of NSZ measured values

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

4. Function declarations in C

All versions of the KK library (*KK_FX80E.DLL* for Windows 32-bit, *KK_Library_64.*dll for Windows 64-bit, *libkk_fx80e.so* for Linux) export the same functions.

All functions have the calling convention **stdcall**:

Parameter transfer from right to left

Return values in registers

Called function clears the stack. Therefore, no variable argument lists are possible.

The following integer sizes are required:

- int: 4 bytes
- short: 2 bytes
- char: 1 byte
- bool: 1 byte

For all *char** parameters it is assumed that 1024 bytes buffer are provided.

Note:

KK library versions with the **cdecl** calling convention are available for Python on Linux systems: libkk_library_32_cdecl.so and libkk_library_64_cdecl.so.

4.1 Create multiple connections

- int **CreateMultiSource**(void);

4.2 List available interfaces

- int **Multi_EnumerateDevices**(char *Names, unsigned char EnumFlags);
- char * **Multi_GetEnumerateDevicesErrorMsg**(void);
- int **Multi_GetHostAndIPs**(char *HostName, char *IPaddr, char * ErrorMsg);

Note:

Multi_GetHostAndIPs char parameters: 80 bytes buffer are enough

4.3 Path definitions

- char * **Multi_GetOutputPath**(int ID);
- char * **Multi_SetOutputPath**(int ID, char *path);

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

4.4 Debug protocol

- char * **Multi_Debug**(int ID, bool DbgOn, char *DbgID);
- int **Multi_DebugFlags**(int ID, bool ReportLog, bool LowLevelLog);
- int **Multi_DebugLogLimit**(int ID, unsigned char LogType, unsigned int aSize);
- char* **Multi_DebugGetFilename**(intl D);

4.5 Info queries

- char * **Multi_GetDLLVersion**(void);
- int **Multi_GetBufferAmount**(int ID);
- int **Multi_GetTransmitBufferAmount**(int ID);
- unsigned char **Multi_GetUserID**(int ID);
- bool **Multi_IsFileDevice**(int ID);
- bool **Multi_IsSerialDevice**(int ID);
- int **Multi_GetFirmwareVersion**(int ID);
- bool **Multi_HasFRAM**(int ID);

4.6 Open and close connection

- int **Multi_OpenConnection**(int ID, char *Connection, bool BlockingIO);
- void **Multi_CloseConnection**(int ID);

4.7 Read reports

- int **Multi_SetDecimalSeparator**(int ID, char Separator);
- int **Multi_SetNSZ**(int ID, int aNSZ);
- int **Multi_GetReport**(int ID, char *Data);

4.8 Send commands

- unsigned int **Multi_GetPendingCmdsCount**(int ID);
- int **Multi_SetCommandLimit**(int ID, unsigned int Limit);
- int **Multi_SendCommand**(int ID, char *Command, int Len);
- int **Multi_RemoteLogin**(int ID, unsigned int Password, char *err);

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

4.9 Local TCP server

- int **Multi_StartTcpServer**(int ID, unsigned short *aPort);
- int **Multi_StopTcpServer**(int ID);
- char * **Multi_GetTcpServerError**(int ID);
- void **Multi_TcpReportLog**(int ID, char *Data, int logType)

4.10 Connection to a TCP server at LOG level

- int **Multi_OpenTcpLog**(int ID, char *IpPort, char *Mode);
- int **Multi_OpenTcpLogTime**(int ID, char *IpPort, char *Mode, char *Format);
- int **Multi_OpenTcpLogType**(int ID, char *IpPort, int LogType, char *Format);
- void **Multi_CloseTcpLog**(int ID);
- int **Multi_GetTcpLog**(int ID, char *Data);

4.11 Send AppData to a TCP server

- int **Multi_TcpAppData**(int ID: Integer; char *Data);

4.12 Generate test data

- int **Multi_StartSaveBinaryData**(int ID, char *DbgID);
- int **Multi_StopSaveBinaryData**(int ID);
- int **Multi_StartSaveReportData**(int ID, char *DbgID);
- int **Multi_StopSaveReportData**(int ID);

4.13 NSZ Calibriation

- int **Multi_SetNSZCalibrationData**(int ID, char *Data);

4.14 FHR settings

- int **Multi_ReadFHRData**(int ID);
- int **Multi_SetFHRData**(int ID, char *Data);

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

5. Function declarations in Java

For the integration of the KK-Library in Java, the wrapper class *KK_Library* is available in the package **brendes.jna-1.7.jar**. Furthermore the packages **jna-4.2.2.jar and jna-platform-4.2.2.jar** are necessary.

The package **brendes.jna-1.8.jar** contains javadoc and source classes and is not described here.

If you need more information, please contact K+K GmbH.

Dependencies of Java packages and KK library versions:

Package	**Minimum Version KK-Library**
brendes.jna-1.0.jar	KK_FX80E Version 16.04
brendes.jna-1.1.jar	KK_FX80E Version 16.05
brendes.jna-1.2.jar	KK_FX80E Version 16.10
brendes.jna-1.3.jar	KK_FX80E Version 18.00
brendes.jna-1.4.jar	KK_FX80E, KK_Library_64 Version 18.01.10
brendes.jna-1.5.jar	KK_FX80E, KK_Library_64 Version 19.00.02
brendes.jna-1.6.jar	19.01.02
brendes.jna-1.7.jar	19.02.00
brendes.jna-1.8.jar	19.03.00

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de

6. Function declarations in Python

The *NativeLib* wrapper class in the **kklib** package is available for integrating the KK library in Python.

There are also example programs **LogReceiver** and **ReportReceiver** that demonstrate the use of *NativeLib*.

All Python modules are provided as source text, without a further description here.

If you need more information, please contact K+K GmbH.

Dependencie of Python Package kklib and KK library versions:

Version Package kklib	Minimum version KK-Library
1.0	18.0
1.0.1	18.01.05
1.0.2	18.01.06
1.0.3	18.01.10
1.0.4	18.02.04
1.1.0	19.00.02
1.2	19.01.02
1.3	19.02.00
1.4	19.03.00

K+K Messtechnik GmbH, St-Wendel-Str. 12, D-38116 Braunschweig, Tel: +49(0)531/501436, kplusk@t-online.de