FORTH DIMENSION DISPLAYS

A subsidiary of Kopin Corporation

CommLib API

for CommLib V1.x

Document No.: AN0026 Revision: AB Issued: 20th February 2017

Forth Dimension Displays Limited, 7 St. David's Drive, St. David's Business Park, Dalgety Bay,
Fife, KY11 9NB, United Kingdom. Telephone: +44 (0) 1383 827 950 Fax: +44 (0) 1383 827 951
E-mail: info@forthdd.com Website: www.forthdd.com

Contents

1 Preface

1.1 Purpose

This document describes CommLib API.

1.2 Scope

This document describes the functions provided by CommLib V1.x. It is not specific to any particular board-level product – Details of functions contained in board-specific libraries may be found in the appropriate API document for your device.

If you have a different version of CommLib, or require the API document for your device, please contact Forth Dimension Displays (ForthDD) for the appropriate Application Note.

1.3 Disclaimer

Please refer to the disclaimer on the last page prior to reading this document.

1.4 Related Documents

Table 1-1: Related Documents

No.	Document No.	Description

These documents are referenced throughout this document using square brackets.

1.5 Third-party Names and Trademarks

- Microsoft Windows is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.
- Java is either a trademark or registered trademark of Oracle Corporation, Inc.
- All other trademarks are the property of their respective holders.

Forth Dimension Displays Limited is independent of and not endorsed by any of the above organisations.

2 Introduction

The CommLib API provides core functionality for communicating with ForthDD board-level products over USB and RS-232/RS-485.

CommLib is written entirely in C and compiled as a static library for Windows. It is designed for use in applications written in C/C++, and includes Java Native Interface (JNI) wrapper functions for easy integration with the Java programming language.

The library is not intended for standalone use, but instead is linked with additional functions (RPC layer) to form a device-specific library which is provided in the form of a dynamic-link library (DLL) for customer use.

Figure 2-1 presents the internal structure of the device-specific library and CommLib.

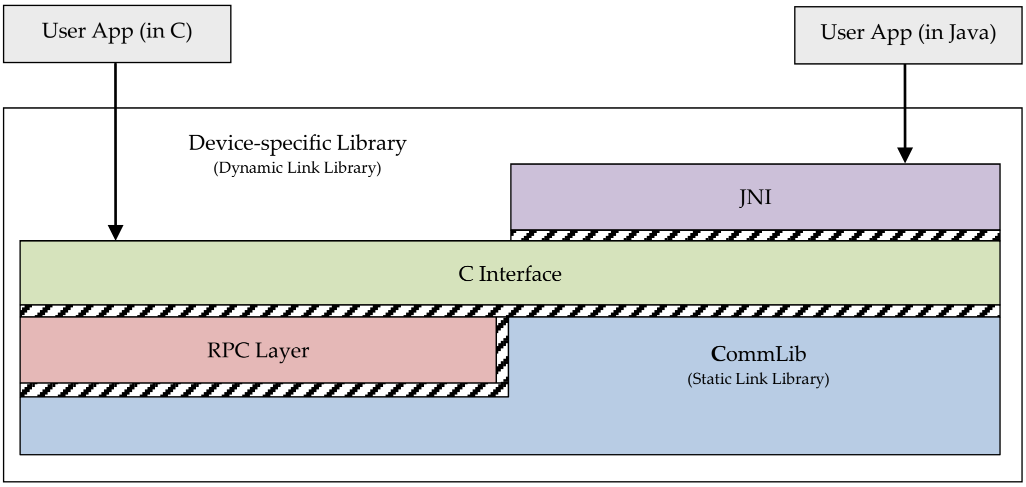

Figure 2-1: Library structure

3 Data Types

The following C pre-processor definitions appear in "core/api/types.h" and are used throughout the CommLib API.

```
#ifndef BOOL
#define BOOL uint8_t
#endif

#ifndef FALSE
#define FALSE     0
#endif

#ifndef TRUE
#define TRUE      1
#endif

#ifndef NULL
#ifdef __cplusplus
#define NULL    0
#else
#define NULL    ((void *)0)
#endif
#endif
```

The data structure below is defined in "core/api/device.h" and used by enumeration functions to produce a linked-list of RS-232/RS-485 and USB (virtual RS-232 / HID / WinUSB) devices attached to the system.

```
typedef struct Dev
{
    char *id;
    struct Dev *next;
} Dev, *DevPtr;
```

The *id* field is a pointer to a null-terminated string. The content of this string is dependent on the host operating system and the interface type, as shown in Table 3-1.
The *next* field contains a pointer to the next Dev structure in the linked-list, or NULL if the end of the list has been reached.

Note that the interface types and operating systems listed may not be supported by all ForthDD products. Refer to the device-specific API document for details on which interfaces and operating systems are supported by your device.
Please contact ForthDD if you require the API document for your device.

Interface	O.S	Contents of *id* field	Example
RS-232/ RS-485	Windows	Port name	`COM1`
	Linux	Dev path	`/dev/ttyS0`
USB (virtual RS-232)	Windows	Port name : S/N	`COM14:1234`
	Linux	Dev path : S/N	`/dev/19ec_0900_1234:1234`
USB-HID	Windows	Dev path : S/N	`\\?\hid#vid_19ec&pid_0301#1234#:1234`
	Linux	Dev path : S/N	`/dev/19ec_0901_1234:1234`
WinUSB	Windows	Dev path : S/N	`\\?\usb#vid_19ec&pid_0403#1234#:1234`
	Linux	Dev path : S/N	`/dev/19ec_0503_1234:1234`

Table 3-1: Contents of *id* field in *Dev* structure

4 Return Values

Unless stated otherwise, all C functions in CommLib have a return type of `FDD_RESULT` (defined in `"core/api/types.h"`) to indicate the result of the function call. `FDD_RESULT` is an enumerated type that contains the members listed in Table 4-1.

`FDD_SUCCESS`	The function returned without error.
`FDD_MEM_INDEX_OUT_OF_BOUNDS`	Parameter value is out of bounds.
`FDD_MEM_NULL_POINTER`	Parameter is null where null is not permitted.
`FDD_MEM_ALLOC_FAILED`	Failed to allocate memory.
`FDD_DEV_SET_TIMEOUT_FAILED`	Timeout value cannot be set.
`FDD_DEV_SET_BAUDRATE_FAILED`	Baud rate cannot be set. This may be caused by an invalid Baud rate value.
`FDD_DEV_OPEN_FAILED`	Unable to open device.
`FDD_DEV_NOT_OPEN`	Operation failed because no device is currently open.
`FDD_DEV_ALREADY_OPEN`	Another device is currently open.
`FDD_DEV_NOT_FOUND`	The specified device was not found.
`FDD_DEV_ACCESS_DENIED`	Device is in use by another application, or user has insufficient privilege to access the device.
`FDD_DEV_READ_FAILED`	Read operation failed. This may be caused by unexpected disconnection of the device.
`FDD_DEV_WRITE_FAILED`	Write operation failed. This may be caused by unexpected disconnection of the device.
`FDD_DEV_TIMEOUT`	Read or write operation timed-out. This may be caused by unexpected disconnection of the device.
`FDD_DEV_RESYNC_FAILED`	Board failed to respond at connection time. This may occur when attempting to connect to physical RS-232 port where no device is present.
`FDD_SLAVE_INVALID_PACKET`	An invalid packet type was returned by the board.
`FDD_SLAVE_UNEXPECTED_PACKET`	A valid, but unexpected packet was returned by the board.
`FDD_SLAVE_ERROR`	The board encountered an error. This may occur if the board receives a malformed or corrupted data packet.
`FDD_SLAVE_EXCEPTION`	The board responded with an exception. This usually indicates that an RPC function received an invalid parameter. The exception message can be retrieved by calling `FDD_ExcGetMsg()`

Table 4-1: Members of `FDD_RESULT`

5 Reference

5.1 FDD_LibGetVersion

Description	Retrieve the version number of CommLib.

Synopsis	`#include "core/api/library.h"`
	`FDD_RESULT FDD_LibGetVersion(char *version);`

Parameters	
`version`	Pointer to a buffer that will receive the version number as a null-terminated string of no more than `LIB_VERSION_MAX_LEN` (defined in `"core/api/library.h"`) bytes in length. The format of the string is given below, and each part of the version number may have a value between 0 and 65535. `major.minor.revision.build\0`
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native String libGetVersion();`

5.2 FDD_ExcGetMsg

Description	Retrieve the previous exception message.

Synopsis	`#include "core/api/exception.h"`
	`FDD_RESULT FDD_ExcGetMsg(const char **msg);`

Parameters	
msg	Address of a `const char *` through which the null-terminated exception message can be read. The exception message is updated when the communications library receives an EXC packet from the slave device (indicated by an RPC function returning `FDD_SLAVE_EXCEPTION`). If no exception message has been received then *msg* will point to an empty string.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Java	No Java equivalent is implemented. Surround code with try-catch block, and call getMessage( ) method of any caught BoardException (subclass of AbstractException) to obtain the exception message string.

5.3 FDD_DevEnumerateComPorts

Description	Obtain a list of serial ports (physical and virtual) present on the host system.

Synopsis	`#include "core/api/device.h"`
	`FDD_RESULT FDD_DevEnumerateComPorts(DevPtr *devList, uint16_t *devCount);`

Parameters	
`devList`	Address of a pointer to the first node in a linked-list that holds the name of all serial ports. The function will set *`devList`* to `NULL` if no matching devices are found. This parameter may be set to `NULL` by the caller if it is not required (for example, if `FDD_DevGetFirst` / `FDD_DevGetNext` are used instead). The caller does not need to allocate memory for use by this function.
`devCount`	Pointer to a variable that receives the number of serial ports found. This parameter may be set to `NULL` by the caller if it is not required.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Notes	Refer to Section 3 for details of the `DevPtr` structure. Device IDs may also be obtained by calling `FDD_DevGetFirst` and `FDD_DevGetNext` immediately after this function. In this case, the parameters *`devList`* and *`devCount`* above may be set to `NULL`.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native String[] devEnumerateComPorts() throws AbstractException;`

5.4 FDD_DevEnumerateHID

Description	Obtain a list of ForthDD USB human interface devices (HID) attached to the host system.

Synopsis	`#include "core/api/device.h"`
	`FDD_RESULT FDD_DevEnumerateHID(uint16_t` *`vid`*`, uint16_t` *`pid`*`, DevPtr *`*`devList`*`, uint16_t *`*`devCount`*`);`

Parameters	
`vid`	USB vendor ID of the HID. This value will be defined in the device-specific API if your device supports the HID interface.
`pid`	USB product ID of the HID. This value will be defined in the device-specific API if your device supports the HID interface.
`devList`	Address of a pointer to the first node in a linked-list that holds the device path and serial number of all devices with the specified vendor and product ID. The function will set *`devList`* to `NULL` if no such devices are found. This parameter may be set to `NULL` by the caller if it is not required (for example, if `FDD_DevGetFirst` / `FDD_DevGetNext` are used instead). The caller does not need to allocate memory for use by this function.
`devCount`	Pointer to a variable that receives the number of matching USB devices found. This parameter may be set to `NULL` by the caller if it is not required.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Notes	Refer to Section 3 for details of the `DevPtr` structure. Device IDs may also be obtained by calling `FDD_DevGetFirst` and `FDD_DevGetNext` immediately after this function. In this case, the parameters *`devList`* and *`devCount`* above may be set to `NULL`.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native String[] devEnumerateHID(int vid, int pid) throws AbstractException;`

5.5 FDD_DevEnumerateWinUSB

Description	Obtain a list of ForthDD WinUSB devices attached to the host system.

Synopsis	`#include "core/api/device.h"`
	`FDD_RESULT FDD_DevEnumerateWinUSB(const char *guid, DevPtr *devList, uint16_t *devCount);`

Parameters	
`guid`	GUID of the WinUSB device. This value will be defined in the device-specific API if your device supports the WinUSB interface.
`devList`	Address of a pointer to the first node in a linked-list that holds the device path and serial number of all devices with the specified GUID. The function will set `devList` to `NULL` if no such devices are found. This parameter may be set to `NULL` by the caller if it is not required (for example, if `FDD_DevGetFirst` / `FDD_DevGetNext` are used instead). The caller does not need to allocate memory for use by this function.
`devCount`	Pointer to a variable that receives the number of matching USB devices found. This parameter may be set to `NULL` by the caller if it is not required.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Notes	Refer to Section 3 for details of the `DevPtr` structure. Device IDs may also be obtained by calling `FDD_DevGetFirst` and `FDD_DevGetNext` immediately after this function. In this case, the parameters `devList` and `devCount` above may be set to `NULL`.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native String[] devEnumerateWinUSB(String guid)`
	`      throws AbstractException;`

5.6 FDD_DevGetFirst

Description	Return the ID of the *first* device found by enumeration
Versions	Introduced in CommLib V1.7

Synopsis	`#include "core/api/device.h"`
	`FDD_RESULT FDD_DevGetFirst(char **`*`pDevId`*`);`

Parameters	
`pDevId`	Address of char pointer (e.g. *`devId`*) that will receive a null-terminated string containing the ID of the *first* device found. The caller does not need to allocate memory for use by this function. *`devId`* will point to null if no device is found.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Notes	Refer to Table 3-1 for details of the device ID string. `FDD_DevGetFirst` must only be called after calling one of the enumeration functions (e.g. `FDD_DevEnumerateWinUSB`). If a device ID is found by `FDD_DevGetFirst`, then `FDD_DevGetNext` may then be called repeatedly until no more devices IDs are found.
Java	Not implemented. Use the required enumeration function instead.

5.7 FDD_DevGetNext

Description	Return the ID of the *next* device found by enumeration
Versions	Introduced in CommLib V1.7

Synopsis	`#include "core/api/device.h"`
	`FDD_RESULT FDD_DevGetNext(char **`*`pDevId`*`);`

Parameters	
`pDevId`	Address of a char pointer (e.g. *`devId`*) that will receive a null-terminated string containing the ID of the *next* device found. The caller does not need to allocate memory for use by this function. *`devId`* will point to null when no more devices are found.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Notes	See the notes section of 5.6.
Java	Not implemented. Use the required enumeration function instead.

5.8 FDD_DevOpenComPort

Description	Open a connection to the specified physical or virtual serial port.

Synopsis	`#include "core/api/device.h"`
	`FDD_RESULT FDD_DevOpenComPort(const char *portName, uint16_t timeout, uint32_t baudRate, BOOL doResync);`

Parameters	
`portName`	The name or device path of the physical or virtual serial port to be opened, expressed as a null-terminated string.
`timeout`	Timeout value (expressed in milliseconds) for I/O operations.
`baudRate`	The Baud rate for communication with the device. Refer to the device-specific API document for your device.
`doResync`	Initiates a brief exchange of data between the host and board at connection time in order to flush the communication buffers. It is recommended that this parameter is always set to TRUE, except when communicating via RS-485, in which case it should always be set to FALSE.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native void devOpenComPort(String portName, int timeout, int baudRate, boolean doResync) throws AbstractException;`

5.9 FDD_DevOpenHID

Description	Open a connection to the specified USB HID.

Synopsis	`#include "core/api/device.h"`
	`FDD_RESULT FDD_DevOpenHID(const char *devPath, uint16_t timeout);`

Parameters	
`devPath`	Device path portion of the string returned by `FDD_DevEnumerateHID`, expressed as a null-terminated string.
`timeout`	Timeout value (expressed in milliseconds) for I/O operations.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native void devOpenHID(String devPath, int timeout) throws AbstractException;`

5.10 FDD_DevOpenWinUSB

Description	Open a connection to the specified WinUSB device.

Synopsis	`#include "core/api/device.h"`
	`FDD_RESULT FDD_DevOpenWinUSB(const char *devPath, uint16_t timeout);`

Parameters	
`devPath`	Device path portion of the string returned by `FDD_DevEnumerateWinUSB`, expressed as a null-terminated string.
`timeout`	Timeout value (expressed in milliseconds) for I/O operations.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native void devOpenWinUSB(String devPath, int timeout) throws AbstractException;`

5.11 FDD_DevSetTimeout

Description	Set the I/O timeout value for the currently opened device.

Synopsis	`#include "core/api/device.h"`
	`FDD_RESULT FDD_DevSetTimeout(uint16_t timeout);`

Parameters	
`timeout`	Timeout value (expressed in milliseconds) for I/O operations.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native void devSetTimeout(int timeout) throws AbstractException;`

5.12 FDD_DevGetTimeout

Description	Get the I/O timeout value for the currently opened device.

Synopsis	`#include "core/api/device.h"`
	`FDD_RESULT FDD_DevGetTimeout(uint16_t *timeout);`

Parameters	
`timeout`	Pointer to a variable that receives the timeout value (expressed in milliseconds) for I/O operations.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native int devGetTimeout() throws AbstractException;`

5.13 FDD_DevClose

Description	Close the connection to the currently opened device.

Synopsis	`#include "core/api/device.h"`
	`FDD_RESULT FDD_DevClose(void);`

Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native void devClose() throws AbstractException;`

5.14 FDD_FlashRead

Description	Read data from the Flash buffer.

Synopsis	`#include "core/api/flash.h"`
	`FDD_RESULT FDD_FlashRead(void *`*`buf`*`, uint16_t `*`offset`*`, uint16_t `*`len`*`);`

Parameters	
`buf`	Pointer to a buffer that will receive data read from the Flash buffer.
`offset`	Offset within the Flash buffer from which the data is read.
`len`	The number of bytes to read.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native byte[] flashRead(int offset, int len) throws AbstractException;`

5.15 FDD_FlashWrite

Description	Write data to the Flash buffer.

Synopsis	`#include "core/api/flash.h"`
	`FDD_RESULT FDD_FlashWrite(const void *buf, uint16_t offset, uint16_t len);`

Parameters	
buf	Constant pointer to a buffer containing data to write to the Flash buffer.
offset	Offset within the Flash buffer to which the data is written.
len	The number of bytes to write.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.core.CommLib`
	`public static native void flashWrite(byte[] buf, int offset) throws AbstractException;`

6 Java Exceptions

6.1 AbstractException

Description	Abstract parent class from which other CommLib exception classes are derived.	
Package	`com.forthdd.commlib.exceptions`	
Declaration	`public abstract class AbstractException extends Exception`	
Methods	`public int getCode()`	Returns the code of the error that caused the exception. The code is the integer value of the corresponding `FDD_RESULT` enum member.
	`Public String getMessage()`	Inherited from java.lang.Throwable. Returns the detail message of the exception.
	`disconnectRecommended()`	Some exceptions are fatal to the connection, while others are not as serious. If this method returns true, it is recommended that the connection is dropped.

6.2 BoardException

Description	Thrown if the following error occurs: `FDD_SLAVE_EXCEPTION`
Package	`com.forthdd.commlib.exceptions`
Declaration	`public class BoardException extends AbstractException`

6.3 CommException

Description	Thrown if one of the following communication errors occur:
	`FDD_DEV_SET_TIMEOUT_FAILED`
	`FDD_DEV_SET_BAUDRATE_FAILED`
	`FDD_DEV_OPEN_FAILED`
	`FDD_DEV_NOT_OPEN`
	`FDD_DEV_ALREADY_OPEN`
	`FDD_DEV_NOT_FOUND`
	`FDD_DEV_ACCESS_DENIED`
	`FDD_DEV_READ_FAILED`
	`FDD_DEV_WRITE_FAILED`
	`FDD_DEV_RESYNC_FAILED`
	`FDD_SLAVE_ERROR`

Package	`com.forthdd.commlib.exceptions`
Declaration	`public class CommException extends AbstractException`

6.4 MemoryException

Description	Thrown if one of the following memory errors occur:
	`FDD_MEM_INDEX_OUT_OF_BOUNDS`
	`FDD_MEM_NULL_POINTER`
	`FDD_MEM_ALLOC_FAILED`

Package	`com.forthdd.commlib.exceptions`
Declaration	`public class MemoryException extends AbstractException`

6.5 PacketException

Description	Thrown if one of the following packet errors occur:
	`FDD_SLAVE_INVALID_PACKET`
	`FDD_SLAVE_UNEXPECTED_PACKET`

Package	`com.forthdd.commlib.exceptions`
Declaration	`public class PacketException extends AbstractException`

6.6 TimeoutException

Description	Thrown if one of the following timeout errors occur: `FDD_DEV_TIMEOUT`
Package	`com.forthdd.commlib.exceptions`
Declaration	`public class TimeoutException extends AbstractException`

7 Revision History

DCN	Originator	Date	Description of Change	New rev.
N/A	CL	11th Nov 2014	• First Draft.	Draft 0.1
D00242	CL	18th Feb 2015	• First Release. Change to applicable CommLib version referenced on cover page and throughout document.	Rev AA
D00310	CL	20th Feb 2017	• Sections 5.3, 5.4 and 5.5: Added notes. • Sections 5.6 and 5.7: New convenience functions for obtaining list of connected devices.	Rev AB

8 Contact Details

For further details please contact:

Forth Dimension Displays Limited
7 St. David's Drive
St. David's Business Park
Dalgety Bay
Fife
KY11 9NB
United Kingdom

Tel: +44 (0) 1383 827 950
Fax: +44 (0) 1383 827 951

Email: info@forthdd.com
Web: www.forthdd.com

All performance figures and other data contained in this document must be confirmed in writing before they become applicable to any tender, order or contract. The Company reserves the right to make alterations or amendments to the information in this document and/or product specifications at its discretion. Forth Dimension Displays Limited does not accept liability for any loss or damage arising from the use of any information or particulars in this application note or from any incorrect use of the product by unauthorised personnel. All maintenance and service of the products must be authorised by Forth Dimension Displays Limited. All reasonable skill and care has been taken in compiling this document. Whilst every effort has been made to ensure the accuracy of the information set out herein no warranty confirming such should be taken as having been given (expressly or implied). No freedom to use patents or other intellectual property rights is implied by the publication of this document. Forth Dimension Displays Limited, 7 St. David's Drive, St. David's Business Park, Dalgety Bay, Fife, KY11 9NB, UK. Forth Dimension Displays Limited is a company registered in England and Wales, registered number: 5220480.