R4CommLib API

for R4CommLib V1.x

Document No.: AN0029 Revision: AC Issued: 30th May 2018

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 the API of the Forth Dimension Displays (ForthDD) R4 Communications Library (R4CommLib) for the SXGA-3DM (R4) and SXGA-R12 (R12).

1.2 Scope

This document describes the Remote Procedure Call (RPC) functions that are exported by R4CommLib V1.x for use with the R4 and R12.

If you have a different version of R4CommLib, please contact Forth Dimension Displays (ForthDD) for the appropriate Application Note.

Refer to *CommLib API* [1] for details of CommLib core functions exported by R4CommLib.

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
1	AN0026	CommLib API
2	UM0015	MetroCon SXGA User Guide
	UM0032	MetroCon SXGA-R12 User Guide

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 R4CommLib API provides programmatic control of the SXGA–3DM (R4) and SXGA-R12 (R12) systems over USB or serial port connections. The communication interfaces supported by each platform is indicated in Table 2-1.

The library is written entirely in C and compiled as a Dynamic-Link Library (DLL) for Windows. It is designed for use in applications written in C/C++ and includes Java Native Interface (JNI) bindings for easy integration with the Java programming language. A list of supported operating systems is shown in Table 2-2.

The serial port must be configured through R4CommLib at connection time with the settings shown in Table 2-3.

Table 2-1: Supported communication interfaces

Platform	Supported Interfaces			
	USB (HID)	**USB (WinUSB)**	**Serial Port (RS-232)**	**Serial Port (RS-485)**
R4	*1	*2	✓	✗
R12	✗	✓	✓	✓

*1. Provided by micro code V2.x only.
*2. Provided by micro code V3.x, or later.

Table 2-2: Supported operating systems

Windows Vista (32 and 64-bit)
Windows 7 (32 and 64-bit)
Windows 8 (32 and 64-bit)
Windows 10 (32 and 64-bit)

Table 2-3: Serial port settings

Standard	Settings
RS-232	115200 / 8-N-1
RS-485	256000 / 8-N-1

2.1 Library Structure

The library is comprised of two main components:

- **CommLib**

  CommLib provides an API for communicating with ForthDD board-level products. It includes functions for enumerating devices, establishing a connection, and reading/writing data. CommLib functions are identified by the name prefix 'FDD_', and are detailed in *CommLib API* [1].

- **RPC Layer**

  The RPC Layer exports a collection of functions that provide access to R4 or R12 specific functionality, such as setting/retrieving LED brightness, changing a Running Order, and monitoring board voltages. These functions are identified by the name prefix 'R4_' or 'R12_', and are detailed in this document.

Figure 2-1 shows the structure of R4CommLib.

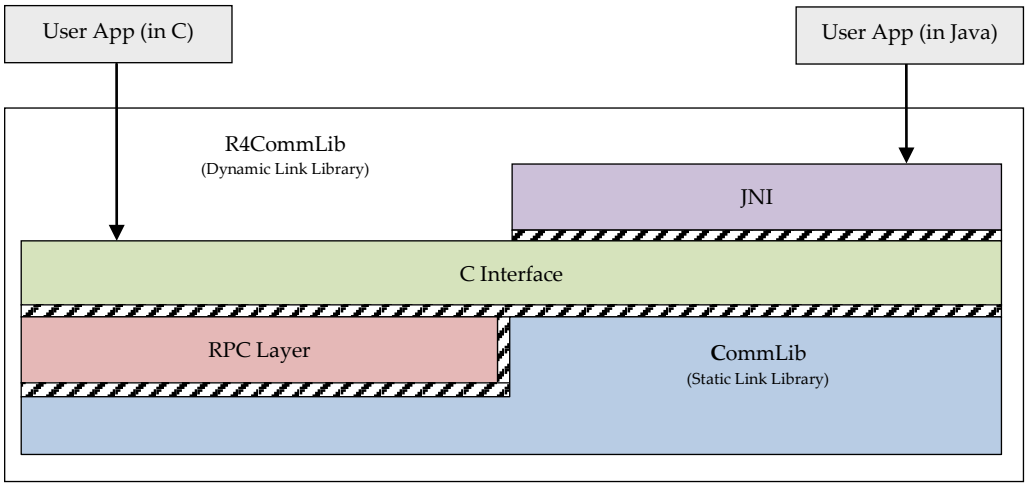

Figure 2-1: Library structure

2.2 API Package

An API package for R4CommLib can be found on the software distribution CD included with starter kits or one with each product shipment. The package contains all the binaries, header files and example projects necessary to begin development with R4CommLib. The contents of the API package are shown in Table 2-4.

Table 2-4: API package contents

Item	Relative path	File name	Description
Core headers	inc/core/api	device.h	Declares functions for enumerating, and opening/closing device connections.
	inc/core/api	exception.h	Declares a function for retrieving exception messages from the device.
	inc/core/api	library.h	Declares a function for retrieving the CommLib version number.
	inc/core/api	types.h	Defines data types used by the library.
R4 headers	inc/r4/api	device.h	Declares a function for retrieving progress of remote operation, and defines various device identifiers and parameters for the R4 and R12.
	inc/r4/api	flash.h	Declares functions for reading/writing on-board Flash, and defines various Flash addresses and parameters.
	inc/r4/api	library.h	Declares a function for retrieving the R4CommLib version number.
	inc/r4/api	rpc.h	Declares functions for invoking Remote Procedure Calls.
Binaries	lib/windows_x86	R4CommLib.dll	32-bit Dynamic Link Library.
	lib/windows_x86	R4CommLib.lib	Stub file for static linking of 32-bit DLL.
	lib/windows_x64	R4CommLib.dll	64-bit Dynamic Link Library.
	lib/windows_x64	R4CommLib.lib	Stub file for static linking of 64-bit DLL.
Visual Studio example projects	load_bitplane		Example to demonstrate the sending of image data to the board.
	toggle_led		Example to demonstrate the changing of the LED brightness

3 Reference

Unless stated otherwise, all API functions have a return type of `FDD_RESULT`. This data type is documented in *CommLib API* [1].

Except where indicated, functions with names that begin with "R4_" can be used with both the R4 and R12, but functions starting with "R12_" can only be used with the R12.

Table 3-1 contains the complete list of R4 and R12 functions, along with the first versions of the library and micro code that provide support for them.

Table 3-1: R4CommLib functions

Function	First supported in version…		
	R4CommLib	R4 Micro Code	R12 Micro Code
R4_LibGetVersion	1.0	N/A	N/A
R12_RpcSysSelectAddr	1.5	-	1.0
R4_RpcSysGetBoardType	1.1	2.1	1.0
R4_RpcSysReboot	1.0	2.1	1.0
R4_RpcSysGetStoredChecksum	1.0	2.1	1.0
R4_RpcSysGetCalculatedChecksum	1.0	2.1	1.0
R4_RpcSysGetBitplaneCount	1.0	2.1	1.0
R4_RpcSysReloadRepertoire	1.0	2.1	1.0
R4_RpcSysGetRepertoireName	1.1	3.1	1.0
R4_RpcSysGetRepertoireUniqueId	1.5	3.3	1.0
R4_RpcSysSaveSettings	1.0	2.1	1.0
R4_RpcSysGetDaughterboardType	1.0	2.1	1.0
R4_RpcSysGetADC	1.0	2.1	1.0
R4_RpcSysGetBoardID	1.0	2.1	1.0
R4_RpcSysGetDisplayType	1.0	2.1	1.0
R4_RpcSysGetDisplayTemp	1.0	2.1	1.0
R4_RpcSysGetSerialNum	1.0	2.1	1.0
R4_RpcMicroGetCodeTimestamp	1.0	2.1	1.0
R4_RpcMicroGetCodeVersion	1.0	2.1	1.0
R4_RpcFlashEraseBlock	1.0	2.1	1.0
R4_RpcRoGetCount	1.0	2.1	1.0
R4_RpcRoGetSelected	1.0	2.1	1.0
R4_RpcRoGetDefault	1.0	2.1	1.0
R4_RpcRoSetSelected	1.0	2.1	1.0
R4_RpcRoSetDefault	1.0	2.1	1.0
R4_RpcRoGetActivationType	1.0	2.1	1.0
R4_RpcRoGetActivationState	1.0	2.1	1.0

R4_RpcRoActivate	1.0	2.1	1.0
R4_RpcRoDeactivate	1.0	2.1	1.0
R4_RpcRoGetName	1.1	3.1	1.0
R4_RpcLedSet	1.0	2.1	1.0
R4_RpcLedGet	1.0	2.1	1.0
R4_RpcFlipTpSet	1.0	2.1	1.0
R4_RpcFlipTpGet	1.0	2.1	1.0
R4_RpcMaintLedSet	1.0	3.1	1.0
R4_RpcMaintLedGet	1.0	3.1	1.0
R4_DevGetProgress	1.3	2.1	1.0
R4_FlashRead	1.3	2.1	1.0
R4_FlashWrite	1.3	2.1	1.0
R4_FlashBurn	1.3	2.1	1.0
R4_FlashGrab	1.3	2.1	1.0
R4_RpcM137CurrentLimitSet	1.7	-	1.0
R4_RpcM137CurrentLimitGet	1.7	-	1.0
R4_RpcM137TemperatureLimitSet	1.7	-	1.0
R4_RpcM137TemperatureLimitGet	1.7	-	1.0

3.1 R4_LibGetVersion

Description	Retrieve the version number of R4CommLib.
Synopsis	`#include "r4/api/library.h"` `FDD_RESULT R4_LibGetVersion(char *verStr, uint8_t maxLen);`
Parameters	
verStr	Pointer to an array that will receive the version number as a null-terminated string. The format of the string is as follows: `major.minor.revision.build\0`
maxLen	Maximum number of bytes to copy to the array. If this value is less than the length of the version number string (including the null terminator) then the destination string will not be null-terminated.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native String libGetVersion() throws AbstractException;`

3.2 R12_RpcSysSelectAddr

Description	Specify the board address for communicating over RS-485. This function is for use with the **R12 only**.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R12_RpcSysSelectAddr(uint8_t rs485Addr);`
Parameters	
`rs485Addr`	The RS-485 address (0 – 7).
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code. `FDD_DEV_TIMEOUT` will be returned if no board connected to the RS-485 bus is assigned the specified address.
Notes	This function must be called immediately after initiating communication via RS-485, before calling any other RPC. Refer to the *MetroCon User Guide* [2] for details on how to assign the RS-485 board address and other important information regarding the use of RS-485.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native void rpcSysSelectAddr(int rs485Addr) throws AbstractException;`

3.3 R4_RpcSysGetBoardType

<table>
<tr><td>Description</td><td>Determine the board type.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Synopsis</td><td><pre>#include "r4/api/rpc.h"
FDD_RESULT R4_RpcSysGetBoardType(uint8_t *boardType);</pre></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Parameters</td></tr>
<tr><td>boardType</td><td>Pointer to an unsigned 8-bit integer that will indicate the board type:
<table>
<tr><th>Value</th><th>Board</th></tr>
<tr><td>0x01</td><td>SXGA-3DM (R4)</td></tr>
<tr><td>0x02</td><td>Reserved</td></tr>
<tr><td>0x03</td><td>SXGA-R12 (R12)</td></tr>
</table></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Return value</td><td><code>FDD_SUCCESS</code> if successful, otherwise the return value is an <code>FDD_RESULT</code> error code.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Java</td><td><pre>com.forthdd.commlib.r4.R4CommLib
public static native byte rpcSysGetBoardType()
throws AbstractException;</pre></td></tr>
</table>

3.4 R4_RpcSysReboot

Description	Restart the on-board micro.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysReboot(void);`
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Notes	The reboot process may take over 10 seconds to complete as the micro must first disable the USB connection and allow sufficient time for the host to register the device removal before performing a soft reset.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native void rpcSysReboot()`
	`throws AbstractException;`

3.5 R4_RpcSysGetStoredChecksum

Description	Retrieve the checksum for the specified bitplane, as stored in the bitplane catalogue.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysGetStoredChecksum(uint16_t bpIndex, uint32_t *bpChecksum);`
Parameters	
bpIndex	The index number of the bitplane (0 – 767).
bpChecksum	Pointer to an unsigned 32-bit integer that will receive the stored checksum.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Notes	Refer also to R4_RpcSysGetCalculatedChecksum. The stored and calculated checksums can be compared to verify the integrity of image data.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native int rpcSysGetStoredChecksum(int bpIndex) throws AbstractException;`

3.6 R4_RpcSysGetCalculatedChecksum

Description	Retrieve the checksum for the specified bitplane, as calculated by scanning the external flash contents.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysGetCalculatedChecksum(uint16_t bpIndex, uint32_t *bpChecksum);`
Parameters	
`bpIndex`	The index number of the bitplane (0 – 767).
`bpChecksum`	Pointer to an unsigned 32-bit integer that will receive the calculated checksum.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Notes	Refer also to R4_RpcSysGetStoredChecksum. The stored and calculated checksums can be compared to verify the integrity of image data.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native int rpcSysGetCalculatedChecksum(int bpIndex) throws AbstractException;`

3.7 R4_RpcSysGetBitplaneCount

Description	Retrieve the number of bitplanes stored in flash.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysGetBitplaneCount(uint32_t *bpCount);`
Parameters	
bpCount	Pointer to an unsigned 32-bit integer that will receive the number of bitplanes.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native int rpcSysGetBitplaneCount()`
	`throws AbstractException;`

3.8 R4_RpcSysReloadRepertoire

Description	Initiate reloading of the repertoire.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysReloadRepertoire(void);`
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Notes	Immediately after invoking this RPC, the user must make repeated calls to `R4_DevGetProgress()` until the function returns a progress value of 100 to indicate completion. This must be done whether or not the user wishes to use the progress value for other purposes (e.g. Updating a progress indicator).

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native void rpcSysReloadRepertoire()`
	`throws AbstractException;`

3.9 R4_RpcSysGetRepertoireName

Description	Retrieve the name of the repertoire loaded on the board.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysGetRepertoireName(char *repName, uint8_t maxLen);`
repName	Pointer to an array that will receive the repertoire name as a null-terminated string of no more than 255 bytes in length (including the null terminator).
maxLen	Maximum number of bytes to copy to the array. If this value is less than the length of the repertoire name (including the null terminator) then the destination string will not be null-terminated.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native String rpcSysGetRepertoireName() throws AbstractException;`

3.10 R4_RpcSysGetRepertoireUniqueId

Description	Retrieve the unique identifier of the repertoire loaded on the board.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysGetRepertoireUniqueId(char *uniqueId, uint8_t maxLen);`
`uniqueId`	Pointer to an array that will receive the unique identifier as a null-terminated string.
`maxLen`	Maximum number of bytes to copy to the array. If this value is less than the length of the unique identifier (including the null terminator) then the destination string will not be null-terminated.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native String rpcSysGetRepertoireUniqueId() throws AbstractException;`

3.11 R4_RpcSysSaveSettings

Description	Write global and per-sequence RAM settings to flash.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysSaveSettings(void);`
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native void rpcSysSaveSettings()`
	`throws AbstractException;`

3.12 R4_RpcSysGetDaughterboardType

<table>
<tr><td>Description</td><td>Determine the LED daughterboard type (if any).</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Synopsis</td><td><code>#include "r4/api/rpc.h"</code>
<code>FDD_RESULT R4_RpcSysGetDaughterboardType(uint8_t *dbType);</code></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Parameters</td></tr>
<tr><td>dbType</td><td>Pointer to an unsigned 8-bit integer that identifies the daughterboard type.
<table>
<tr><th>Value</th><th>Daughterboard</th></tr>
<tr><td>0x00</td><td>No LED daughterboard</td></tr>
<tr><td>0x01</td><td>M118 Rev A</td></tr>
<tr><td>0x02</td><td>M118 Rev B</td></tr>
<tr><td>0x11</td><td>M137 Rev A</td></tr>
</table></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Return value</td><td><code>FDD_SUCCESS</code> if successful, otherwise the return value is an <code>FDD_RESULT</code> error code.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Java</td><td><code>com.forthdd.commlib.r4.R4CommLib</code>
<code>public static native byte rpcSysGetDaughterboardType()
throws AbstractException;</code></td></tr>
</table>

3.13 R4_RpcSysGetADC

Description	Read the value from the specified Analog to Digital Converter (ADC) channel. The ADC provides monitoring of the various power supply lines on the board.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysGetADC(uint8_t adcChannel, uint16_t *adcValue);`
Parameters	
adcChannel	The ADC channel number. Channel numbers and corresponding supplies are shown in the tables below.
adcValue	Pointer to an unsigned 16-bit integer that returns the ADC value. Each channel corresponds to a different voltage or current measurement, as indicated in Table 3-2 for R4 and Table 3-3 for R12. The actual value may be obtained by the following formula: $actualValue = ((adcValue + offset) \times reference) / (maxValue \times factor)$ Where, $maxValue = 1023$ and $reference = 3.0$ The values of *offset* and *factor* are dependent on the supply being measured.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code. Returns `FDD_SLAVE_EXCEPTION` if: • Channel number is out of range [0..7]
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native int rpcSysGetADC(int channel) throws AbstractException;`

Table 3-2: R4 ADC channels

Channel	Description	Nominal	Units	Offset	Factor
0	2V5	2.5	V	0	1.0
1	1V8	1.8	V	0	1.0
2	1V2	1.2	V	0	1.0
3	Unused	-	-	-	-
4	12V	12.0	V	0	5/33
5	5V	5.0	V	0	0.5
6	3V3	3.3	V	0	0.5
7	Unused	-	-	-	-

Table 3-3: R12 ADC channels

Channel	Description	Nominal	Units	Offset	Factor
0	PGOOD	3.3	V	0	0.5
1	12V	12	V	0	127 / 1036
2	+5V_DISP	5	V	0	0.5
3	CURRENT	0.23	A	-188	105123 / 50000
4 – 11	Reserved	-	-		-

3.14 R4_RpcSysGetBoardID

Description	Determine the board revision.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysGetBoardID(uint8_t *boardId);`
Parameters	
`boardId`	Pointer to an unsigned 8-bit integer that will receive the 4-bit ID of the main board. The following table shows the possible values for *`boardId`* and the corresponding hardware revision.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native byte rpcSysGetBoardID()`
	`throws AbstractException;`

Value	Revision
0x0	A
0x1	B
0x2	C
0x3	D
0x4	E
0x5	F

3.15 R4_RpcSysGetDisplayType

<table>
<tr><td>Description</td><td>Determine the type of display that is connected (if any).</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Synopsis</td><td><code>#include "r4/api/rpc.h"</code>
<code>FDD_RESULT R4_RpcSysGetDisplayType(uint8_t *dispType);</code></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Parameters</td></tr>
<tr><td>dispType</td><td>Pointer to an unsigned 8-bit integer that will indicate the display type.
<table>
<tr><th>Value</th><th>Display</th><th>Resolution</th><th>Compatible Platforms</th></tr>
<tr><td>0x00</td><td>None</td><td>N/A</td><td>All</td></tr>
<tr><td>0x01</td><td>M149/M249</td><td>SXGA (1280 × 1024)</td><td>1, 3</td></tr>
<tr><td>0x02</td><td>M108</td><td>WXGA (1280 × 768)</td><td>3</td></tr>
<tr><td>0x03</td><td>M150</td><td>QXGA (2048 × 1536)</td><td>2</td></tr>
<tr><td>0x04</td><td>M449/M549</td><td>SXGA</td><td>1, 3</td></tr>
<tr><td>0xFF</td><td>Unknown</td><td>N/A</td><td>All</td></tr>
</table>
1. SXGA-3DM (R4)
2. QXGA-3DM (R11)
3. SXGA-R12 (R12)</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Return value</td><td><code>FDD_SUCCESS</code> if successful, otherwise the return value is an <code>FDD_RESULT</code> error code.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Java</td><td><code>com.forthdd.commlib.r4.R4CommLib</code>
<code>public static native byte rpcSysGetDisplayType()
throws AbstractException;</code></td></tr>
</table>

3.16 R4_RpcSysGetDisplayTemp

Description	Retrieve the output from the display temperature sensor, if it has one.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysGetDisplayTemp(uint16_t *dispTemp);`
Parameters	
dispTemp	Pointer to an unsigned 16-bit integer to receive the display temperature.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native int rpcSysGetDisplayTemp()`
	`throws AbstractException;`

3.17 R4_RpcSysGetSerialNum

Description	Retrieve the serial number of the board.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcSysGetSerialNum(uint32_t *serialNum);`
Parameters	
`serialNum`	Pointer to an unsigned 32-bit integer that will receive the serial number.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native int rpcSysGetSerialNum()`
	`throws AbstractException;`

3.18 R4_RpcMicroGetCodeTimestamp

Description	Retrieve the micro code timestamp.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcMicroGetCodeTimestamp(char *timestamp, uint8_t maxLen);`
Parameters	
timestamp	Pointer to an array that will receive the timestamp as a null-terminated string. The format of the timestamp is as follows: *Www Mmm dd hh:mm:ss yyyy\0* Where Www is the weekday, Mmm the month (in letters), dd the day of the month, hh:mm:ss the time, and yyyy the year.
maxLen	Maximum number of bytes to copy to the array. If this value is less than or equal to the length of the timestamp string then the destination string will not be null-terminated.
Return value	FDD_SUCCESS if successful, otherwise the return value is an FDD_RESULT error code.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native String rpcMicroGetCodeTimestamp() throws AbstractException;`

3.19 R4_RpcMicroGetCodeVersion

<table>
<tr><td>Description</td><td>Retrieve the micro code version number.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Synopsis</td><td><code>#include "r4/api/rpc.h"</code>
<code>FDD_RESULT R4_RpcMicroGetCodeVersion(uint16_t *version);</code></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Parameters</td></tr>
<tr><td>version</td><td>Pointer to an array of four 16-bit unsigned integers that will receive the version number.
The array values are interpreted as follows:
<table>
<tr><th>Index</th><th>Constituent</th></tr>
<tr><td>0</td><td>Build Number</td></tr>
<tr><td>1</td><td>SVN Revision</td></tr>
<tr><td>2</td><td>Minor Version</td></tr>
<tr><td>3</td><td>Major Version</td></tr>
</table></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Return value</td><td><code>FDD_SUCCESS</code> if successful, otherwise the return value is an <code>FDD_RESULT</code> error code.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Java</td><td><code>com.forthdd.commlib.r4.R4CommLib</code>
<code>public static native byte[] rpcMicroGetCodeVersion()
throws AbstractException;</code></td></tr>
</table>

3.20 R4_RpcFlashEraseBlock

Description	Erase the flash memory block that contains the specified page.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcFlashEraseBlock(uint32_t page);`
Parameters	
page	Any flash page that resides in the block that is to be erased.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native void rpcFlashEraseBlock(int page) throws AbstractException;`

3.21 R4_RpcRoGetCount

Description	Retrieve the number of running orders.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcRoGetCount(uint16_t *roCount);`
Parameters	
roCount	Pointer to an unsigned 16-bit integer that will receive the running order count.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native int rpcRoGetCount()`
	`throws AbstractException;`

3.22 R4_RpcRoGetSelected

Description	Retrieve the index of the currently selected running order.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcRoGetSelected(uint16_t *roIndex);`
Parameters	
roIndex	Pointer to an unsigned 16-bit integer that will receive the index of the currently selected running order.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native int rpcRoGetSelected()`
	`throws AbstractException;`

3.23 R4_RpcRoGetDefault

Description	Retrieve the index of the power-on default running order.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcRoGetDefault(uint16_t *roIndex);`
Parameters	
roIndex	Pointer to an unsigned 16-bit integer that will receive the index of the power-on default running order.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native int rpcRoGetDefault()`
	`throws AbstractException;`

3.24 R4_RpcRoSetSelected

Description	Change to the specified running order.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcRoSetSelected(uint16_t roIndex);`
Parameters	
roIndex	The index of the running order that is to be selected.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native void rpcRoSetSelected(int roIndex) throws AbstractException;`

3.25 R4_RpcRoSetDefault

Description	Change the power-on default running order.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcRoSetDefault(uint16_t roIndex);`
Parameters	
roIndex	The index of the running order that is to be set as the power-on default.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native void rpcRoSetDefault(int roIndex) throws AbstractException;`

3.26 R4_RpcRoGetActivationType

<table>
<tr><td>Description</td><td>Retrieve the activation type of the currently selected running order.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Synopsis</td><td><code>#include "r4/api/rpc.h"</code>
<code>FDD_RESULT R4_RpcRoGetActivationType(uint8_t *actType);</code></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Parameters</td></tr>
<tr><td>actType</td><td>Pointer to an unsigned 8-bit integer that will indicate the activation type of the currently selected running order.
<table>
<tr><th>Value</th><th>Activation Type</th></tr>
<tr><td>0x01</td><td>Immediate</td></tr>
<tr><td>0x02</td><td>Software</td></tr>
<tr><td>0x04</td><td>Hardware</td></tr>
</table></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Return value</td><td><code>FDD_SUCCESS</code> if successful, otherwise the return value is an <code>FDD_RESULT</code> error code.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Java</td><td><code>com.forthdd.commlib.r4.R4CommLib</code>
<code>public static native byte rpcRoGetActivationType()
throws AbstractException;</code></td></tr>
</table>

3.27 R4_RpcRoGetActivationState

<table>
<tr><td>Description</td><td>Retrieve the activation state of the repertoire/running order.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Synopsis</td><td><code>#include "r4/api/rpc.h"</code>
<code>FDD_RESULT R4_RpcRoGetActivationState(uint8_t *actState);</code></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Parameters</td></tr>
<tr><td>actState</td><td>Pointer to an unsigned 8-bit integer that will indicate the activation state of the repertoire/running order.
<table>
<tr><th>Value</th><th>Activation State</th><th>Description</th></tr>
<tr><td>0x50</td><td>RLD</td><td>Repertoire loading *</td></tr>
<tr><td>0x51</td><td>STA</td><td>Starting *</td></tr>
<tr><td>0x52</td><td>MSW</td><td>Maintenance – Software deactivated</td></tr>
<tr><td>0x53</td><td>MHD</td><td>Maintenance – Hardware and Software deactivated</td></tr>
<tr><td>0x54</td><td>MHW</td><td>Maintenance – Hardware deactivated</td></tr>
<tr><td>0x55</td><td>PAC</td><td>Activating *</td></tr>
<tr><td>0x56</td><td>ACT</td><td>Active</td></tr>
<tr><td>0x57</td><td>NRP</td><td>No Repertoire available</td></tr>
</table>
* These states are transitional. All other states are stable.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Return value</td><td><code>FDD_SUCCESS</code> if successful, otherwise the return value is an <code>FDD_RESULT</code> error code.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Java</td><td><code>com.forthdd.commlib.r4.R4CommLib</code>
<code>public static native byte rpcRoGetActivationState()
throws AbstractException;</code></td></tr>
</table>

3.28 R4_RpcRoActivate

Description	Activate the currently selected running order.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcRoActivate(void);`
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native void rpcRoActivate()`
	`throws AbstractException;`

3.29 R4_RpcRoDeactivate

Description	Deactivate the currently selected running order.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcRoDeactivate(void);`
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native void rpcRoDeactivate()`
	`throws AbstractException;`

3.30 R4_RpcRoGetName

Description	Retrieve the name of the specified running order.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcRoGetName(uint16_t roIndex, char *roName, uint8_t maxLen);`
roIndex	The index of the running order.
roName	Pointer to an array that will receive the running order name as a null-terminated string.
maxLen	Maximum number of bytes to copy to the array. If this value is less than the length of the running order name (including the null terminator) then the destination string will not be null-terminated.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native String rpcRoGetName(int roIndex) throws AbstractException;`

3.31 R4_RpcLedSet

Description	Set the LED brightness value.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcLedSet(uint8_t ledValue);`
Parameters	
`ledValue`	The new LED brightness value (0 – 255).
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native void rpcLedSet(int ledValue)`
	`throws AbstractException;`

3.32 R4_RpcLedGet

Description	Retrieve the LED brightness value.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcLedGet(uint8_t *ledValue);`
Parameters	
`ledValue`	Pointer to an unsigned 8-bit integer that will indicate the LED brightness value.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native int rpcLedGet()`
	`throws AbstractException;`

3.33 R4_RpcFlipTpSet

<table>
<tr><td>Description</td><td>Control test pattern and image flip.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Synopsis</td><td><code>#include "r4/api/rpc.h"</code>
<code>FDD_RESULT R4_RpcFlipTpSet(uint8_t flipTpValue);</code></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Parameters</td></tr>
<tr><td><code>flipTpValue</code></td><td>The flip and test pattern status value:
<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>7 –6</td><td>Reserved.</td></tr>
<tr><td>5</td><td>Reserved.</td></tr>
<tr><td>4</td><td>Test pattern enable.</td></tr>
<tr><td>3 – 2</td><td>Select test pattern.

00 ⇒ Test Bars 1
01 ⇒ Black
10 ⇒ White
11 ⇒ Test Bars 2</td></tr>
<tr><td>1</td><td>0 ⇒ Normal
1 ⇒ Horizontal flip</td></tr>
<tr><td>0</td><td>0 ⇒ Normal
1 ⇒ Vertical flip</td></tr>
</table></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Return value</td><td><code>FDD_SUCCESS</code> if successful, otherwise the return value is an <code>FDD_RESULT</code> error code.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Java</td><td><code>com.forthdd.commlib.r4.R4CommLib</code>
<code>public static native void rpcFlipTpSet(byte flipTpValue)
throws AbstractException;</code></td></tr>
</table>

3.34 R4_RpcFlipTpGet

Description	Retrieve the horizontal and vertical flip value.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcFlipTpGet(uint8_t *flipTpValue);`
Parameters	
`flipTpValue`	Pointer to an unsigned 8-bit integer that will receive the flip and test pattern status value. See `R4_RpcFlipTpSet` for possible values.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native byte rpcFlipTpGet()`
	`throws AbstractException;`

3.35 R4_RpcMaintLedSet

<table>
<tr><td>Description</td><td>Control whether or not LED illumination is enabled during maintenance mode.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Synopsis</td><td><code>#include "r4/api/rpc.h"</code>
<code>FDD_RESULT R4_RpcMaintLedSet(BOOL ledEnable);</code></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Parameters</td></tr>
<tr><td>ledEnable</td><td><code>BOOL</code> value that sets whether or not to enable LED during maintenance mode:
<table><tr><td>0 (FALSE)</td><td>Disable LED</td></tr><tr><td>1 (TRUE)</td><td>Enable LED</td></tr></table></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Return value</td><td><code>FDD_SUCCESS</code> if successful, otherwise the return value is an <code>FDD_RESULT</code> error code.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Java</td><td><code>com.forthdd.commlib.r4.R4CommLib</code>
<code>public static native void rpcMaintLedSet(boolean ledEnable)
throws AbstractException;</code></td></tr>
</table>

3.36 R4_RpcMaintLedGet

Description	Indicate whether or not LED illumination is enabled during maintenance mode.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcMaintLedGet(BOOL *ledEnable);`
Parameters	
`ledEnable`	Pointer to a `BOOL` value that will indicate whether or not LED is enabled during maintenance mode: 0 (FALSE) — LED is disabled 1 (TRUE) — LED is enabled
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native boolean rpcMaintLedGet()`
	`throws AbstractException;`

3.37 R4_DevGetProgress

Description	Get the progress of the current board operation.
Synopsis	`#include "r4/api/device.h"` `FDD_RESULT R4_DevGetProgress(uint8_t *pro);`
Parameters	
pro	Pointer to a `uint8_t` value that receives the current progress expressed as a percentage.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native int devGetProgress()`
	`throws AbstractException;`

3.38 R4_FlashRead

Description	Read data from the Flash buffer.
Synopsis	`#include "r4/api/flash.h"` `FDD_RESULT R4_FlashRead(void *buf, uint16_t offset, uint16_t len);`
Parameters	
buf	Pointer to a user buffer that will receive data from the Flash buffer.
offset	Offset within the Flash buffer from which to begin reading.
len	Number of bytes to read from the Flash buffer.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code. Returns `FDD_MEM_INDEX_OUT_OF_BOUNDS` if: • If the sum of *offset* and *len* exceed the size of the Flash buffer (2048 bytes)
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native byte[] flashRead(int offset, int len) throws AbstractException;`

3.39 R4_FlashWrite

Description	Write data to the Flash buffer.
Synopsis	`#include "r4/api/flash.h"` `FDD_RESULT R4_FlashWrite(const void *buf, uint16_t offset, uint16_t len);`
Parameters	
buf	Pointer to a user buffer containing data to be written to the Flash buffer.
offset	Offset within the Flash buffer at which to begin writing.
len	Number of bytes to write to the Flash buffer.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code. Returns `FDD_MEM_INDEX_OUT_OF_BOUNDS` if: • If the sum of *offset* and *len* exceed the size of the Flash buffer (2048 bytes)
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native void flashWrite(byte[] buf, int offset) throws AbstractException;`

3.40 R4_FlashBurn

Description	Write a single page of data from the Flash buffer in RAM to the Flash device.
Synopsis	`#include "r4/api/flash.h"` `FDD_RESULT R4_FlashBurn(uint32_t page);`
Parameters	
page	Page number in Flash that is to be written.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code. Returns `FDD_MEM_INDEX_OUT_OF_BOUNDS` if: • Page number is located in a reserved area of Flash.
Notes	See section 4 for details of Flash block allocation. The external Flash device is arranged into blocks of 64 pages. A page can only be correctly written if the block in which it resides was previously erased by calling `R4_RpcFlashEraseBlock`. It is only necessary to perform the erase block operation once in order to allow writing of every page within that block.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native void flashBurn(int page)`
	`throws AbstractException;`

3.41 R4_FlashGrab

Description	Read a single page of data from the Flash device into the Flash buffer in RAM.
Synopsis	`#include "r4/api/flash.h"` `FDD_RESULT R4_FlashGrab(uint32_t page);`
Parameters	
page	Page number in Flash that is to be read.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code. Returns `FDD_MEM_INDEX_OUT_OF_BOUNDS` if: • Page number is located in a reserved area of Flash.
Notes	See section 4 for details of Flash block allocation.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native void flashGrab(int page)`
	`throws AbstractException;`

3.42 R4_RpcM137CurrentLimitSet

Description	Set the maximum LED brightness value for an M137 LED Driver Daughterboard. This value will be an upper safety limit, enforced by the micro. Any subsequent call to R4_RpcLedSet (section 3.31) with a value greater than this limit, will be ignored. On systems without an M137, this call has no effect.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcM137CurrentLimitSet(uint8_t maxValue);`
Parameters	
ledValue	The new LED brightness upper safety limit (0 – 255).
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.
Java	`com.forthdd.commlib.r4.R4CommLib` `public static native void rpcM137CurrentLimitSet(int maxValue) throws AbstractException;`

3.43 R4_RpcM137CurrentLimitGet

Description	Retrieve the maximum LED brightness value, as previously set by R4_RpcM137CurrentLimitSet (section 3.42).
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcM137CurrentLimitGet(uint8_t *maxValue);`
Parameters	
ledValue	Pointer to an unsigned 8-bit integer that will indicate the LED brightness upper safety limit (0 – 255).
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native int rpcM137CurrentLimitGet()`
	`throws AbstractException;`

3.44 R4_RpcM137TemperatureLimitSet

Description	Set the maximum temperature (in degrees Celsius) of an LED driven by an M137 LED Driver Daughterboard. If the temperature of the LED, as measured by the attached thermistor, rises above this value, the LEDs will be turned off. On systems without an M137, this call has no effect.
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcM137TemperatureLimitSet(uint8_t maxValue);`
Parameters	
ledValue	The new LED temperature upper safety limit (0 – 255).
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native void rpcM137TemperatureLimitSet(int maxValue)`
	`throws AbstractException;`

3.45 R4_RpcM137TemperatureLimitGet

Description	Retrieve the maximum temperature (in degrees Celsius), as previously set by R4_RpcM137TemperatureLimitSet (section 3.44).
Synopsis	`#include "r4/api/rpc.h"` `FDD_RESULT R4_RpcM137TemperatureLimitGet(uint8_t *maxValue);`
Parameters	
`ledValue`	Pointer to an unsigned 8-bit integer that will indicate the LED maximum temperature.
Return value	`FDD_SUCCESS` if successful, otherwise the return value is an `FDD_RESULT` error code.

Java	`com.forthdd.commlib.r4.R4CommLib`
	`public static native int rpcM137TemperatureLimitGet()`
	`throws AbstractException;`

4 Flash

The system has two flash memories, one internal to the microcontroller and one in a separate external device. The size and organisation of the two devices are as shown in Table 4-1. Sizes are in bytes, unless marked K (Kilobytes) or M (Megabytes).

Table 4-1: Flash devices

Platform	Device	Page size	Pages/Block	Block size	Pages	Blocks	Total size
R4	Internal	256	1	256	1024	1024	256K
R12		256	1	256	512	512	128K
R4 / R12	External	2048	64	128K	131072	2048	256M

The terms *page* and *block* are defined as follows:

page The smallest unit which can be read or written.

block The smallest unit which can be erased.

Space in the devices is allocated as shown in Table 4-2 for R4, and Table 4-3 for R12. Blocks marked 'reserved' are not for user access, and should not be used. The 'User data' area is available for users to store any data they wish.

Table 4-2: R4 Flash block allocation

Start page	Pages	Blocks	Size	Device	Purpose
0x00000000	1024	1024	256K	Internal	Reserved
0x01000000	61440	960	120M	External	Image bitplane data
0X0100F000	256	4	512K	External	Reserved
0x0100F100	3840	60	7.5M	External	User data
0x02000000	131072	2048	256M	External	Test and debug, read only

Table 4-3: R12 Flash block allocation

Start page	Pages	Blocks	Size	Device	Purpose
0x00000000	512	512	128K	Internal	Reserved
0x01000000	61440	960	120M	External	Image bitplane data
0x0100F000	3840	60	7.5M	External	User data
0x0100FF00	256	4	512K	External	Reserved
0x02000000	131072	2048	256M	External	Test and debug, read only

4.1 Flash Programming

Table 4-4 describes the typical steps involved in the programming of Flash pages across multiple blocks on the R4.

Block	Command	Description
1	`R4_RpcFlashEraseBlock(0x01000000)`	Erase Flash block 0x00040000
	`R4_FlashWrite([2048], 0, 2048)`	Write page data to Flash buffer
	`R4_FlashBurn(0x01000000)`	Burn Flash buffer to Flash page 0x01000000
	`R4_FlashWrite([2048], 0, 2048)`	Write page data to Flash buffer
	`R4_FlashBurn(0x01000001)`	Burn Flash buffer to Flash page 0x01000001
	…	
	`R4_FlashWrite([2048], 0, 2048)`	Write page data to Flash buffer
	`R4_FlashBurn(0x0100003F)`	Burn Flash buffer to Flash page 0x0100003F
2	`R4_RpcFlashEraseBlock(0x01000040)`	Erase Flash block 0x00040001
	`R4_FlashWrite([2048], 0, 2048)`	Write page data to Flash buffer
	`R4_FlashBurn(0x01000040)`	Burn Flash buffer to Flash page 0x01000040
	`R4_FlashWrite([2048], 0, 2048)`	Write page data to Flash buffer
	`R4_FlashBurn(0x01000041)`	Burn Flash buffer to Flash page 0x01000041
	…	
	`R4_FlashWrite([2048], 0, 2048)`	Write page data to Flash buffer
	`R4_FlashBurn(0x0100007F)`	Burn Flash buffer to Flash page 0x0100007F

Table 4-4: Flash programming procedure

4.2 Image Data

Each SXGA bitplane requires 80 pages, or 1.25 blocks. Each group of 4 bitplanes occupies 5 complete blocks.

The entire image set can contain up to 768 bitplanes, so it requires 61440 pages, or 960 blocks.

152	153	154	155	156	157	158	159	144	145	146	147	148	149	150	151		8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
312	313	314	315	316	317	318	319	304	305	306	307	308	309	310	311		168	169	170	171	172	173	174	175	160	161	162	163	164	165	166	167

Figure 4-1: Display byte order

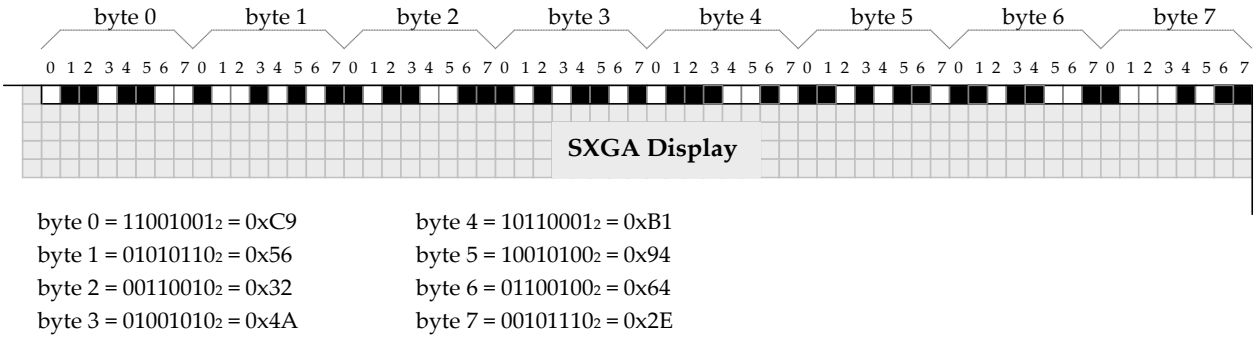

Figure 4-2: Display bit order

As shown in Figure 4-1, the display is filled from right to left and top to bottom. The first 160 bytes from Flash contains the data for the top row, the next 160 bytes contain the data for the second row from the top, and so on.

Each group of 8 bytes appears from left to right, and each byte has the order of its bits reversed such that the least significant bit is on the left (as shown in Figure 4-2). A '1' results in a bright pixel. A '0' results in a dark pixel.

The individual bitplanes comprising an 8-bit image are stored with the least significant bitplane first.

Source code is provided in the 'load_bitplane' example that accompanies R4CommLib.

5 Migrating from MetroLib

R4CommLib is the new communication library for SXGA-3DM (R4) and SXGA-R12 (R12) systems, and is a direct replacement for the MetroLib library. When used in conjunction with the latest version of Micro Code, R4CommLib provides significantly faster device communication and image transfer using WinUSB, as well as providing support for a greater number of Running Orders and new features such as Hardware Assisted Running Orders.

This section outlines the key differences between MetroLib and R4CommLib functions in order to assist with adapting software to use the new library.

Table 5-1 shows the correspondence between MetroLib and R4CommLib data types. The data types FDD_RESULT and BOOL used by R4CommLib are defined in *types.h.*

Please note that in addition to now using the C99 data types shown (as defined in *stdint.h*), some R4CommLib function parameters may have a different bit-width to accommodate changes in the latest version of Micro Code. For example, function `R4_RpcRoSetSelected` accepts a 16-bit unsigned integer whereas the corresponding MetroLib function accepts an 8-bit unsigned integer.

Table 5-1: Data types

	MetroLib	**R4CommLib**
Return type	short int	FDD_RESULT
Parameter types	UBYTE (unsigned char)	uint8_t char
	BOOL (unsigned char)	BOOL (uint8_t)
	UWORD (unsigned short)	uint16_t
	ULONG (unsigned long)	uint32_t

Table 5-2 shows the R4CommLib equivalent for each MetroLib function.

MetroLib functions highlighted in red have been withdrawn, and therefore have no R4CommLib equivalent. R4CommLib functions highlighted in green are new, and therefore have no equivalent in MetroLib. In some cases, more than one equivalent function may exist in R4CommLib.

Careful attention must be paid to MetroLib functions marked with an asterisk (*). The asterisk indicates that the R4CommLib equivalent function has non-trivial differences in either its parameters or its behaviour.

Table 5-2: R4CommLib equivalent functions

MetroLib Function	R4CommLib Equivalent
getLibVersion	N/A
getLibVersion16 *	FDD_LibGetVersion
	R4_LibGetVersion
enumerateUSB *	FDD_DevEnumerateHID
	FDD_DevEnumerateWinUSB
enumerateRS232 *	FDD_DevEnumerateComPorts
openDevice *	FDD_DevOpenComPort
	FDD_DevOpenHID
	FDD_DevOpenWinUSB
closeDevice	FDD_DevClose
setTimeout	FDD_DevSetTimeout
N/A	FDD_DevGetTimeout
setBaudRate	N/A (See FDD_DevOpenComPort)
sendGrab	R4_FlashGrab
sendBurn	R4_FlashBurn
sendRead *	R4_FlashRead
sendWrite *	R4_FlashWrite
getPacket2	N/A
getException *	FDD_ExcGetMsg
isAP	N/A
N/A	R12_RpcSysSelectAddr
N/A	R4_RpcSysGetBoardType
rebootBoard	R4_RpcSysReboot
getTimestamp *	R4_RpcMicroGetCodeTimestamp
getVersion *	R4_RpcMicroGetCodeVersion
eraseBlock	R4_RpcFlashEraseBlock
getNumBitplanes *	R4_RpcSysGetBitplaneCount
N/A	R4_RpcSysGetStoredChecksum
N/A	R4_RpcSysGetCalculatedChecksum
getROCount *	R4_RpcRoGetCount
getSelectedRO *	R4_RpcRoGetSelected
getDefaultRO *	R4_RpcRoGetDefault
setSelectedRO *	R4_RpcRoSetSelected

setDefaultRO *	R4_RpcRoSetDefault
N/A	R4_RpcRoGetName
getActivationType	R4_RpcRoGetActivationType
getActivationState	R4_RpcRoGetActivationState
activate	R4_RpcRoActivate
deactivate	R4_RpcRoDeactivate
N/A	R4_RpcSysGetRepertoireName
N/A	R4_RpcSysGetRepertoireUniqueId
reloadRepertoire	R4_RpcSysReloadRepertoire
saveSettings	R4_RpcSysSaveSettings
setLED	R4_RpcLedSet
getLED	R4_RpcLedGet
N/A	R4_RpcMaintLedSet
N/A	R4_RpcMaintLedGet
setFlipTP *	R4_RpcFlipTpSet
getFlipTP	R4_RpcFlipTpGet
getDaughterboardType	R4_RpcSysGetDaughterboardType
getBoardID	R4_RpcSysGetBoardID
getDisplayType	R4_RpcSysGetDisplayType
getSerialNum	R4_RpcSysGetSerialNum
N/A	R4_RpcSysGetADC
N/A	R4_RpcM137CurrentLimitSet
N/A	R4_RpcM137CurrentLimitGet
N/A	R4_RpcM137TemperatureLimitSet
N/A	R4_RpcM137TemperatureLimitGet

6 Revision History

DCN	Originator	Date	Description of Change	New rev.
D00251	CL	23rd Apr 2015	• First release	AA
D00307	CL	20th Feb 2017	• Table 2-2: Removed Windows XP and added Windows 10. • R4_RpcSysGetSerialNum: Corrected function name. • R4_RpcRoGetActivationState: Newly documented function. • R4_DevGetProgress: Corrected return type of Java method. • R4_FlashRead: Corrected parameters in Java method. • Table 5-2: Added R4_RpcRoGetActivationState	AB
D00343	CL / PBH	30th May 2018	• Updates throughout for SXGA-R12. • Section 2: Added tables 2-1 and 2-3. • Section 3: Added R12_RpcSysSelectAddr Added R4_RpcSysGetRepertoireUniqueId Added R4_RpcSysGetDisplayTemp Added R4_RpcM137CurrentLimitSet Added R4_RpcM137CurrentLimitGet Added R4_RpcM137TemperatureLimitSet Added R4_RpcM137TemperatureLimitGet Added SXGA-R12 to table in R4_RpcSysGetBoardType Updated table in R4_RpcSysGetDaughterboardType Updated R4_RpcSysGetADC and added tables 3-1 and 3-2 Updated table in R4_RpcSysGetADC Updated table in R4_RpcSysGetBoardID Updated table in R4_RpcSysGetDisplayType Updated table in R4_RpcRoGetActivationState Updated table in R4_RpcFlipTpSet • Section 4: Updated table 4-1 Added table 4-3	AC

7 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