User's Guide
DLPU039–September 2016

DLP® Discovery™ 4100 Development Kit API Programmer’s Guide

This manual describes the use of the application programming interface (API) for the DLP® Discovery™ 4100 Development Kit evaluation module (EVM). The development kit combines hardware, software, firmware, and documentation to form a stand-alone platform for use in developing and testing applications designed for use with the Texas Instruments DLP Discovery 4100 Development Kit.

Contents

List of Figures

1 Notational Conventions

This document uses the following conventions:

The graphical user interface is referred to as GUI.

Program listings, program examples, and interactive displays are shown as a special typeface. Some examples use a **bold** version of the special typeface for emphasis. Interactive displays also use a **bold** version of the special typeface to distinguish between commands that you enter from items that the system displays (such as prompts, command output, error messages, and so forth).

Here is a sample program listing:

```
0011 0005 0001 .field 1, 2
0012 0005 0003 .field 3, 4
0013 0005 0006 .field 6, 3
0014 0006 .even
```

In syntax descriptions, the instruction, command, or directive is in a **bold** typeface, and parameters are in an *italic* typeface. Portions of the syntax that are **bold** should be entered as shown; portions of syntax that are in *italics* describe the type of information that should be entered. Syntax that is entered on a command line is centered. Syntax that is used in a text file is left justified.

Square brackets ([example]) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets. Unless the square brackets are in a **bold** typeface, do not enter the brackets themselves.

2 Overview

This manual provides:

- A general description of the DLP Discovery 4100 Development Kit Board API
- An overview of the ActiveX™ interface commands
- Programming Examples

3 Terms and Definitions

API — Application programming interface

Compliment Data — Inverts all incoming image data bits (1's to 0's and 0's to 1's)

D4100 — DLP Discovery 4100 Development Kit (EVM)

DLPC410 — DLPC410 DLP Digital Controller

DMD — Digital micromirror device

GUI — Graphical user interface

North/South Flip — Flips the DMD image data top to bottom (vertically but not horizontally)

USB — Universal serial bus

Variable Types:

NOTE: All variable types are assumed to be unsigned unless otherwise specified.

Char — An 8-bit value used as a text character

Int — An Integer value

Long — 32-bit value

Short — 16-bit value

Signed — An integer in the range between -2^{n-1} and +2^{n-1} -1 : where n is the number of bits in the representation. For example, if n=8, then this is a number between -128 and +127.

Unsigned — An integer in the range between 0 and +2^{n} - 1 : where n is the number of bits in the representation. For example, if n=8, then this is a number between 0 and +255.

LPCTSTR — **L**ong **P**ointer to a **C**onstant **T**CHAR **STR**Ing

UCHAR — A Char used as an 8-bit unsigned integer

*** —** denotes a pointer to a value (example: Char* is a pointer to the Char)

4 API Overview

The DLP Discovery 4100 Development Kit API provides a complete function library to support custom programming of DLP Discovery 4100 Development Kit control applications. Windows 7, 8.x, and 10, as well as both 32-bit and 64-bit Windows development platforms are supported. The API provides control of the DLP Discovery 4100 Development Kit hardware via the USB port by interfacing with an ActiveX control or directly to the D4100_usb.dll.

The ActiveX control is designed to transmit data, send commands, set flags and values, and read values and the state of flags from the DLP D4100 APPS_FPGA via the USB interface. The API provides the capability to control all functions of the DLPC410 controller as well as control the image load and display of the DLP Discovery 4100 Development Kit through the APPS_FPGA.

NOTE: ActiveX™ is now an older technology and may not be compatible with some software. For improved compatibility, direct use of the D4100_usb.dll functions is recommended and typically results in faster execution.

4.1 DMD Image Control

Images are controlled and displayed in blocks on the digital micromirror device (DMD). The DMD is logically divided into blocks: 16 blocks of 48 rows of mirrors with 1024 mirrors in each row for XGA DMDs (DLP7000 & DLP7000UV), and 15 blocks of 72 rows of mirrors with 1920 mirrors in each row for 1080p DMDs (DLP9500 & DLP9500UV). Blocks can be loaded and displayed individually, or as an entire image (all blocks). Refer to the DLPC410 data sheet, DLPS024, for more information.

5 DLP Discovery 4100 Development Kit USB ActiveX API

ActiveX is a Microsoft defined set of technologies that enables software components to interact with one another regardless of the language in which they were created. ActiveX is built on the Component Object Model (COM). ActiveX controls are distributed as .ocx files and are predated by .vbx and .dll files. Refer to http://msdn.microsoft.com/ and search for ActiveX for additional information on ActiveX controls.

The DLP Discovery 4100 Development Kit ActiveX control provides a convenient mechanism for communication between customer developed software applications and the DLP Discovery 4100 Development Kit driver software. The control is distributed in the DDC4100.ocx file and provides an interface for configuration, control and display to the DLP Discovery 4100 Development Kit. When this control is used to build applications in C, C++, or Visual Basic, control of the Discovery board is easily accomplished using the methods documented in this chapter. The following sections describe the ActiveX control features. For information on the us of various values that can be set (for example BlkMd) refer to the DLPC410 data sheet, DLPS024.

NOTE: Before using any ActiveX™ methods or calls, *ConnectDevice* (see Section 5.1.5) must be called to initialize a connection to the DLP Discovery 4100 Development kit USB interface.

5.1 Configuration/Status ActiveX Methods

5.1.1 Void AboutBox( )

Displays an about box showing the ActiveX control version and copyright information.

5.1.2 Short AllowMessages(short value)

Controls the display of error messages by the ActiveX control. TRUE = nonzero = enable messages, FALSE = 0 = disable messages. Default is TRUE.

5.1.3 Short DownloadAppsFPGACode(LPCTSTR FileName)

Loads the FPGA program (.bin format file) specified by *FileName* into the APPS_FPGA of the DLP Discovery 4100 Development Kit. Returns 1 if successful or 0 if unsuccessful. The default file location is in the DLP Discovery 4100 Explorer GUI installation directory and is named *D4100_GUI_FPGA.bin*.

5.1.4 Short GetNumDevices( )

Returns the number of Discovery USB devices connected to the system. They are automatically numbered starting at one. For example, if you have four devices plugged in, GetNumDev( ) will return a 4. To access, these devices you will pass a 1, 2, 3, or a 4 to indicate the respective device to the ConnectDevice function - see Section 5.1.5.

5.1.5 Short ConnectDevice(short DeviceNumber, LPCTSTR FileName)

Initializes a DLP Discovery 4100 Development Kit USB interface and must be called before any other ActiveX functions that use USB communication. *Devicenumber* is used if multiple DLP D4100 boards are connected to the PC, however in most cases only one device will be connected and a 1 will be passed. If the board has not been previously initialized the application APPS_FPGA code (.bin format file) passed as *FileName* will be loaded into the hardware.

Note: This differs from the equivalent .dll function which starts at 0 for numeration.

5.1.6 BOOL IsDeviceAttached( )

Returns a TRUE (1) if a DLP Discovery 4100 Development Kit has been attached using the GetDevice method. Otherwise a FALSE (0) value is returned.

5.1.7 Long GetActivexRev( )

Returns the ActiveX (OCX) revision. Upper 16 bits contain major revision, lower 16 bits contain minor revision.

5.1.8 Long GetDriverRev( )

Returns the USB driver revision. Upper 16 bits contain major revision, lower 16 bits contain minor revision.

5.1.9 Short GetFirmwareRev( )

Returns the USB firmware revision. The high byte contains all the digits before the decimal point and the low byte of the returned value contains all the digits after the decimal point (See Figure 1).

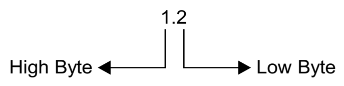

Figure 1. Revision Number Format

5.1.10 Short GetSpeedMode( )

Determines the speed at which the DLP Discovery 4100 Development Kit USB interface is operating. A return value of 1 indicates that the device is operating at high speed (USB 2.0), a value of 0 means it is operating at full speed (USB 1.1).

5.1.11 Long GetDLLRev( )

Returns the version information for this DLL. The high byte contains all the digits before the decimal point and the low byte of the returned value contains all the digits after the decimal point (See Figure 1).

5.1.12 Unsigned int GetFPGARev( )

Returns the version of the firmware running on the D4100 APPS_FPGA. The high byte contains all the digits before the decimal point and the low byte of the returned value contains all the digits after the decimal point (See Figure 1).

5.1.13 short GetDDCVERSION( )

Returns the DLPC410 Version in bits 2, 1, 0.

5.1.14 short GetDMDTYPE( )

Returns DMD types as listed below:

- '0' - for DLP9500 / DLP9500UV [0.95" (Visible / UV) 1080p 2xLVDS Type A DMD]
- '1' - for DLP7000 / DLP7000UV [0.7" (Visible / UV) XGA 2xLVDS Type A DMD]
- ‘16’ – if a DMD is not attached or not recognized by the DLPC410 controller.

5.2 Low Level Control ActiveX Methods

Provides basic DLPC410 control. Any DMD function can be performed with the low level control methods. Low level control methods do not utilize the ActiveX control image buffer, they directly send data and commands over USB.

5.2.1 Short LoadControl( )

Loads all control values to DLPC410 controller, which then loads the values to the DMD as needed. This can be used for all non data transactions such as Micromirror Clocking Pulses (resets) and Clears. Returns 1 if successful or 0 if unsuccessful.

5.2.2 Int LoadData(UCHAR* RowData, long length)

Loads row data into the DLPC410 controller, which is then loaded into the DMD. *RowData* must be in a UCHAR array of size *length* equal to the width of the DMD in pixels. No more than 500 rows can be loaded at a time (96 Kb for 1080p, 51.2 Kb for XGA). To load an entire DMD, call this function multiple times. Returns 1 if successful or 0 if unsuccessful. Row data is displayed in the row in the same order as in *RowData*. Bit 0 will appear as the first pixel in the row and the last bit as the last bit in the row.

5.2.3 Short SetBlkMd (short value)

Sets the BLKMD value in the D4100 APPS_FPGA. Returns 1 if successful or 0 if unsuccessful.

5.2.4 Short GetBlkMd ( )

Gets the BLKMD value from the D4100 APPS_FPGA.

5.2.5 Short SetBlkAd(short value)

Sets the BLKAD value in the D4100 APPS_FPGA. Returns 1 if successful or 0 if unsuccessful.

5.2.6 Short GetBlkAd ( )

Gets the BLKAD value from the D4100 APPS_FPGA.

5.2.7 Short SetRST2BLKZ(short value)

Sets the Micromirror Clocking Pulse to 'by two' mode by setting the Reset Two Blocks flag in the D4100 APPS_FPGA. Active = 0, inactive = 1 (default). Returns 1 if successful or 0 if unsuccessful.

5.2.8 Short GetRST2BLKZ ( )

Gets the value of the Reset Two Blocks flag value from the D4100 APPS_FPGA). 'By two' mode: 0 = active, 1 = inactive.

5.2.9 Short SetRowMd(short value)

Sets the ROWMD value in the D4100 APPS_FPGA. Returns 1 if successful or 0 if unsuccessful.

5.2.10 Short GetRowMd( )

Gets the ROWMD value from the D4100 APPS_FPGA.

5.2.11 Short SetRowAddr(short value)

Sets the ROWAD value in the D4100 APPS_FPGA. Returns 1 if successful or 0 if unsuccessful.

5.2.12 Short GetRowAddr( )

Gets the ROWAD value from the D4100 APPS_FPGA.

5.2.13 short SetCOMPDATA(short value)

Sets the Complement Data flag in the D4100 APPS_FPGA. Active = 1, inactive = 0 (default). Returns 1 if successful or 0 if unsuccessful.

5.2.14 short GetCOMPDATA( )

Gets the current value of the Complement Data flag from the D4100 APPS_FPGA: 1 = active, 0 = inactive.

5.2.15 short SetNSFLIP(short value)

Sets the North/South Flip flag in the D4100 APPS_FPGA. Active = 1, inactive = 0 (default). Returns 1 if successful or 0 if unsuccessful.

5.2.16 short GetNSFLIP( )

Gets the current value of the North/South Flip flag from the D4100 APPS_FPGA: 1 = active, 0 = inactive.

5.2.17 short SetWDT(short value)

Sets the Watch Dog Timer flag in the D4100 APPS_FPGA. Active = 1 (default), inactive = 0 . Returns 1 if successful or 0 if unsuccessful.

5.2.18 short GetWDT( )

Gets the value of the Watch Dog Timer flag from the D4100 APPS_FPGA: 1 = active, 0 = inactive.

5.2.19 Short FloatMirrors( )

Places the DMD in a safe state with the mirrors in the *floated* or flat condition, with no bias applied to the DMD. Returns 1 if successful or 0 if unsuccessful.

5.2.20 short SetPWRFLOAT(short value)

Sets the Power Float flag in the D4100 APPS_FPGA. Active = 1, inactive = 0 (default). Returns 1 if successful or 0 if unsuccessful.

5.2.21 short GetPWRFLOAT( )

Gets the value of the Power Float flag from the D4100 APPS_FPGA: 0 = 1 = active, 0 = inactive.

5.2.22 short SetEXTRESETENBL(short value)

Enables or disables GPIOA.0 as an external Micromirror Clocking Pulse (reset) input. Enabled = 1, disabled = 0 (default).

GPIOA.0 is a 2.5-V CMOS input. When enabled, all software control of the DMD Micromirror Clocking Pulses (resets) is disabled. Reset operation as defined by RST2BLKZ, BLK_MD, and BLK_ADDR will be initiated on rising edge of external Micromirror Clocking Pulse (reset) input. Returns 1 if successful or 0 if unsuccessful.

5.2.23 short GetEXTRESETENBL( )

Gets the current External Reset Enable value from the D4100 APPS_FPGA: 1 = enabled, 0 = disabled.

5.2.24 short SetGPIO(short value)

Sets the output values of GPIOA.2-4 to provide programmable digital outputs. Bits 4, 3, 2 of value control the output state. Bits 7, 6, 5, 1, 0 of value are not used. GPIOA.2-4 are 2.5-V CMOS outputs.

5.2.25 short GetGPIO( )

Returns the values of GPIOA.5-7 as digital inputs. Bits 7, 6, 5 of returned value are the input data. Bits 4, 3, 2, 1, 0 of returned value are not used. GPIOA.5-7 are 2.5-V CMOS inputs.

5.2.26 short GetRESETCOMPLETE(long waittime)

Enables global external Micromirror Clocking Pulses (resets) (see Section 5.2.22) and then loops *waittime* in milliseconds for an external Micromirror Clocking Pulse (reset) trigger event to happen. If external Micromirror Clocking Pulse (reset) occurs in the *waittime* windows this function will return a 1. Set *waittime* to 0 to wait indefinitely.

5.3 DMD Display Operation ActiveX Methods

DMD images may be sourced from an image file or constructed directly by software. Refer to Figure 2 for an illustration of image display methods.

Image files should be 1024 × 768 pixels for XGA DMDs (DLP7000 & DLP7000UV), or 1920 × 1080 for 1080p DMDs (DLP9500 & DLP9500UV). An image file must be reformatted to a format which can be sent to the DMD. This re-formatting first thresholds each pixel using a value of RGB (i.e. If, R, G, or B is ≥ 240 then the value of the pixel in the resulting binary image is set to 1) to convert from a grayscale image to a binary image with 1 bit per pixel. The pixel values are stored in a binary format for DMD loading. Images may be pre-converted and stored to disk file .bin format using ConvertImage or stored to the memory images buffer using LoadImageFileToBuffer. The pre-converted images may be quickly displayed on the DMD.

The converted image data is transferred to an Active Control Internal Memory buffer. The data is transferred from this buffer to the DMD via the USB driver.

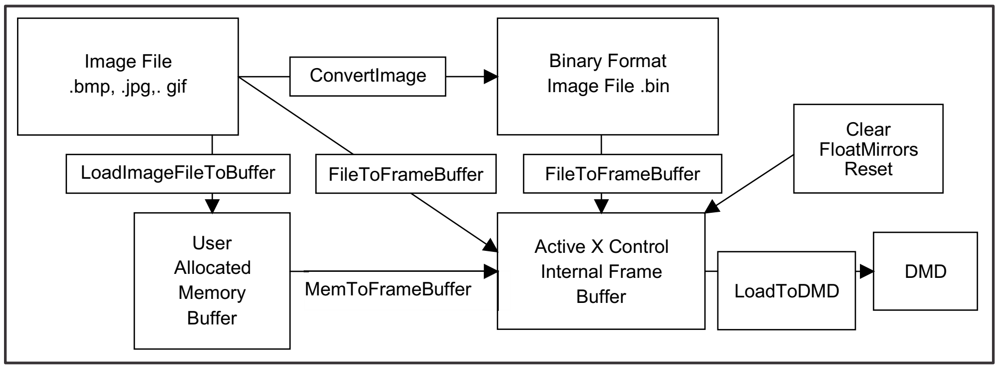
Image File .bmp, .jpg,. gif
ConvertImage
Binary Format Image File .bin
Clear FloatMirrors Reset
LoadImageFileToBuffer
FileToFrameBuffer
FileToFrameBuffer
User Allocated Memory Buffer
MemToFrameBuffer
Active X Control Internal Frame Buffer
LoadToDMD
DMD

Figure 2. Graphical User Interface Layout

5.3.1 Short Clear(short BlockNum, short DoReset)

Clears the contents of an entire block on the DMD. Specify the block to be cleared in *BlockNum*. Block numbers range from 1 to 16, a block number greater than 16 will signal a Global Clear. A Micromirror Clocking Pulse (reset) is executed on the specified blocks if *DoReset* is nonzero. Returns 1 if successful or 0 if unsuccessful.

5.3.2 Short ConvertImage (LPCTSTR SrcFile, LPCTSTR DestFile, short MirrorImage)

Converts a standard bmp, jpg, or gif image file, indicated by the file *SrcFile*, into a binary image file (.bin) which may be sent directly to the display. The binary image is stored into the file indicated by *DestFile*. A mirror image may be created by setting *MirrorImage* to a non-zero value. Returns 1 if successful or 0 if unsuccessful.

A .bin file contains 1 bit per DMD pixel arranged starting with pixel (0,0) and arranged in row, column format. For example, for an XGA DMD the first 1024 bits in the file contain the image data for the first row, the second 1024 bits contains the image data for row 2, and so on.

5.3.3 short SetConversionThreshold(short threshold)

Will set the RGB (8,8,8) to binary (0 or 1) image conversion threshold to a value between 0 and 255. For example, a value of 127 will threshold at a value of RGB (127,127,127).

5.3.4 Short FileToFrameBuffer(LPCTSTR ImageFile, short MirrorImage)

Processes an image file (*ImageFile*) into a binary format which may be sent directly to the display and load the binary image into the ActiveX control image buffer. Setting *MirrorImage* to a non-zero value will instruct the method to construct a mirrored image. The image file must be in the form of a Bitmap file (*.bmp), a JPEG file (*.jpg), a GIF file (*.gif), or a binary file (*.bin). The converted image is stored in the ActiveX control's image buffer. Returns 1 if successful or 0 if unsuccessful.

5.3.5 Short LoadToDMD (short BlockNum, short DoReset)

Loads a block of data from the ActiveX control image buffer to the DMD. Specify the block to be loaded in *BlockNum*. Block numbers range from 1 to 16, a block number greater than 16 will load the entire DMD. Image data is loaded from the ActiveX control’s image buffer. A Micromirror Clocking Pulse (reset) for selected blocks is executed if *DoReset* is nonzero. Returns 1 if successful or 0 if unsuccessful.

5.3.6 Short MemToFrameBuffer (unsigned short* ImageBufferPtr)

Loads the ActiveX control image buffer with a binary image. The binary image may be read directly from a file generated by the ***ConvertImage*** method or may be program generated. Returns 1 if successful or 0 if unsuccessful.

5.3.7 Short LoadImageFileToBuffer(LPCTSTR FileName, unsigned short* ImageBufferPtr, short MirrorImage)

Loads the memory buffer *ImageBuffer* with the desired image. The image is converted to a binary format. The image is read a from a standard bmp, jpg, or gif image file, indicated by *FileName*. Setting *MirrorImage* to a non-zero value will instruct the method to construct a mirrored image. Returns 1 if successful or 0 if unsuccessful.

5.3.8 Short Reset(short BlockNum)

Executes a Micromirror Clocking Pulse (reset) operation for the block specified by BlockNum. Block numbers range from 1 to 16, a block number greater than 16 will signal a Global Reset. Returns 1 if successful or 0 if unsuccessful.

5.4 ActiveX Control Usage Examples

Control sequences for common Discovery operations are presented below in programming language-independent examples:

5.4.1 Open USB Device

The USB device must be attached prior to performing any DMD operations. After connecting, the device will remain open until the calling program ends.

```
DDC4100Ctrl.ConnectDevice(1,C:\usb_main.bin)
// Open channel to device
```

5.4.2 Display Single Image on DMD

A single image can be loaded from a standard image file and displayed on the DMD using the following methods:

```
DDC4100Ctrl.FileToFrameBuffer(ImageFile, MirrorImage)
// Load ActiveX control’s frame buffer from specified standard .bmp, .jpg, .gif,
//    or .bin image file.
DDC4100Ctrl.LoadResetFrame( )
// Write Image to DMD and Micromirror Clocking Pulse (reset)
```

5.4.3 Display Multiple Images on DMD

To maximize display speed, images should be converted and loaded to memory buffer or buffers prior to starting display.

Multiple images are first converted and loaded to memory buffers by repeating this method as needed:

```
DDC_DDC4100Ctrl_Ctrl.LoadImageFileToBuffer(FileName, ImageBuffer, MirrorImage)
// Convert image from specified standard .bmp, .jpg, or .gif image file to a format which
//    can be directly written to the DMD. Store converted file in memory buffer ImageBuffer.
```

An image is displayed from memory by passing a pointer to the memory buffer:

```
DDC4100Ctrl.MemToFrameBuffer(VarPBuffer)
// load ActiveX control’s frame buffer from memory pointer VarPBuffer
DDC4100Ctrl.LoadResetFrame( )
// Write Image to DMD and Micromirror Clocking Pulse (reset)
```

5.4.4 Clear Block on DMD

```
DDC4100Ctrl.Clear(BlockNum,0)
// Clear block without Reset
DDC4100Ctrl.Clear(BlockNum,1)
// Clear block and Reset
DDC4100Ctrl.Reset(BlockNum)
// Reset block
```

5.4.5 Clear Entire DMD Display

```
DDC4100CtrlCtrl.Clear(17,0)
// Global clear without Reset
DDC4100CtrlCtrl.Clear(17,1)
// Global clear and Reset
DDC4100CtrlCtrl.Reset(17)
// Global Micromirror Clocking Pulse (reset)
```

6 DLP Discovery 4100 Development Kit USB DLL API

Dynamic Link Libraries are Microsoft's implementation of shared libraries. They contain functions that are accessed on run time and have a .DLL extension. The DLP Discovery 4100 Development Kit DLL library contains the necessary low level functions as an alternative to the ActiveX control. For information on the us of various values that can be set (for example BlkMd), refer to the DLPC410 data sheet, DLPS024.

6.1 Configuration/Status DLL Functions

6.1.1 int program_FPGA(UCHAR* write_buffer, long write_size, short int DeviceNumber)

Programs the APPS_FPGA. The APPS_FPGA configuration file must first be loaded into a UCHAR (1 byte per element) array. The write_size is in bytes. Program control will remain with the DLL until the board has finished programming the FPGA.

6.1.2 short GetNumDev( )

Returns the number of DLP Discovery 4100 Development kits attached via USB to the system. They will automatically be numbered starting at zero. For example, if you have four devices plugged in, GetNumDev( ) will return a 4. To access these devices you will pass the other dll functions a 0, 1, 2, or a 3 for that respective device.

6.1.3 int GetDescriptor(int*, short DeviceNum);

Returns the USB device descriptor information. It will return the number of transferred bytes if successful, a -1 if the USB device failed to open, and a -2 if the device descriptor request fails.

Integer Array information:

Array[0] = bLength

Array[1] = bDescriptorType

Array[2] = bcdUSB

Array[3] = bDeviceClass

Array[4] = bDeviceSubClass

Array[5] = bDeviceProtocol

Array[6] = bMaxPacketSize0

Array[7] = idVendor

Array[8] = idProduct

Array[9] = bcdDevice

Array[10] = iManufacturer

Array[11] = iProduct

Array[12] = iSerialNumber

Array[13] = bNumConfigurations

6.1.4 unsigned int GetDriverRev(short DeviceNumber)

Returns the USB driver revision. Upper 16 bits contain major revision, lower 16 bits contain minor revision.

6.1.5 unsigned int GetFirmwareRev(short DeviceNumber)

Returns the firmware revision on the USB (Cypress FX2) device. The high byte contains all the digits before the decimal point and the low byte of the returned value contains all the digits after the decimal point (See Figure 1).

6.1.6 short int GetUsbSpeed(short DeviceNumber)

Determines if the device is plugged into a high-speed (USB 2.0) or full speed (USB 1.1) USB port. Returns a 0 if the port is full speed, a 1 if the port is high-speed.

Negative values indicate errors:

- -1 = USB Device failed to open
- -2 = Device Descriptor request failed
- -3 = USB speed value differs from expected (not USB 2.0 or 1.1)

6.1.7 unsigned int GetDLLRev( )

Returns the version information for this DLL. The high byte contains all the digits before the decimal point and the low byte of the returned value contains all the digits after the decimal point (See Figure 1).

6.1.8 unsigned int GetFPGARev(short DeviceNumber)

Returns the version of the firmware running on the D4100 applications FPGA. The high byte contains all the digits before the decimal point and the low byte of the returned value contains all the digits after the decimal point (See Figure 1).

6.1.9 short GetDDCVERSION(short DeviceNumber)

Returns the DLPC410 Version in bits 2,1,0.

6.1.10 short GetDMDTYPE(short DeviceNumber)

Returns the DMD type as listed below:

- '0' - for DLP9500 / DLP9500UV [0.95" (Visible / UV) 1080p 2xLVDS Type A DMD]
- '1' - for DLP7000 / DLP7000UV [0.7" (VisibleF / UV) XGA 2xLVDS Type A DMD]
- ‘16’ – if a DMD is not attached or not recognized by the DLPC410 controller.

6.2 Low Level Control DLL Functions

6.2.1 short LoadControl(short DeviceNumber)

Loads all control flags to DMD. This can be used for all non data transactions such as Micromirror Clocking Pulses (resets), and Clears. Returns 1 if successful or 0 if unsuccessful.

6.2.2 int LoadData(UCHAR* RowData, unsigned int length, short DMDType, short DeviceNumber)

Loads *RowData* into the DMD. Data must be in a UCHAR array of size *length*. The short *DMDType* should be the Type returned by short ***GetDMDTYPE***(*short DeviceNumber*) [see Section 6.1.10]. No more than 500 rows can be loaded at a time (96Kbit for 1080p, 51.2Kbit for XGA). To load an entire DMD call this function multiple times. Returns 1 if successful or 0 if unsuccessful.

6.2.3 short ClearFifos(short DeviceNumber)

Resets hardware receiving FIFO buffers. This should be used to put the device into a known state before sending DMD image data. Returns 1 if successful or 0 if unsuccessful.

6.2.4 short SetBlkMd(short value, short DeviceNumber)

Sets the BLKMD value in the D4100 APPS_FPGA. Returns 1 if successful or 0 if unsuccessful.

6.2.5 short GetBlkMd(short DeviceNumber)

Gets the BLKMD value from the D4100 APPS_FPGA.

6.2.6 short SetBlkAd(short value, short DeviceNumber)

Sets the BLKAD value in the D4100 APPS_FPGA. Returns 1 if successful or 0 if unsuccessful.

6.2.7 short GetBlkAd(short DeviceNumber)

Gets the BLKAD value from the D4100 APPS_FPGA.

6.2.8 short SetRST2BLKZ(short value, short DeviceNumber)

Sets the Micromirror Clocking Pulse to 'by two' mode by setting the Reset Two Blocks flag value in the D4100 APPS_FPGA. Active = 0, inactive = 1 (default). Returns 1 if successful or 0 if unsuccessful.

6.2.9 short GetRST2BLKZ(short DeviceNumber)

Gets the value of the Reset Two Blocks flag value from the D4100 APPS_FPGA). 'By two' mode: 0 = active, 1 = inactive.

6.2.10 short SetRowMd(short value, short DeviceNumber)

Sets the ROWMD value in the D4100 APPS_FPGA. Returns 1 if successful or 0 if unsuccessful.

6.2.11 short GetRowMd(short DeviceNumber)

Gets the ROWMD value from the D4100 APPS_FPGA.

6.2.12 short SetRowAddr(short value, short DeviceNumber)

Sets the ROWAD value in the D4100 APPS_FPGA. Returns 1 if successful or 0 if unsuccessful.

6.2.13 short GetRowAddr(short DeviceNumber)

Gets the ROWAD value from the D4100 APPS_FPGA.

6.2.14 short SetCOMPDATA(short value, short DeviceNumber)

Sets the Complement Data flag in the D4100 APPS_FPGA. Active = 1, inactive = 0 (default). Returns 1 if successful or 0 if unsuccessful.

6.2.15 short GetCOMPDATA(short DeviceNumber)

Gets the current value of the Complement Data flag from the D4100 APPS_FPGA: 1 = active, 0 = inactive.

6.2.16 short SetNSFLIP(short value, short DeviceNumber)

Sets the North/South Flip flag in the D4100 APPS_FPGA. Active = 1, inactive = 0 (default). Returns 1 if successful or 0 if unsuccessful.

6.2.17 short GetNSFLIP(short DeviceNumber)

Gets the current value of the North/South Flip flag from the D4100 APPS_FPGA: 1 = active, 0 = inactive.

6.2.18 short SetWDT(short value, short DeviceNumber)

Sets the Watch Dog Timer flag on the D4100 APPS_FPGA. Active = 1 (default), inactive = 0. Returns 1 if successful or 0 if unsuccessful.

6.2.19 short GetWDT(short DeviceNumber)

Gets the value of the Watch Dog Timer flag from the D4100 APPS_FPGA: 1 = active, 0 = inactive.

6.2.20 short SetPWRFLOAT(short value, short DeviceNumber)

Sets the Power Float flag in the D4100 APPS_FPGA. Active = 1, inactive = 0 (default). Returns 1 if successful or 0 if unsuccessful.

6.2.21 short GetPWRFLOAT(short DeviceNumber)

Gets the value of the Power Float flag from the D4100 APPS_FPGA: 0 = 1 = active, 0 = inactive.

6.2.22 short SetEXTRESETENBL(short value, short DeviceNumber)

Enables or disables GPIOA.0 as an external Micromirror Clocking Pulse (reset) input. Enabled = 1, disabled = 0 (default).

GPIOA.0 is a 2.5-V CMOS input. When enabled, all software control of the DMD Micromirror Clocking Pulses (resets) is disabled. Reset operation as defined by RST2BLKZ, BLK_MD, and BLK_ADDR will be initiated on rising edge of external Micromirror Clocking Pulse (reset) input. Returns 1 if successful or 0 if unsuccessful.

6.2.23 short GetEXTRESETENBL(short DeviceNumber)

Gets the current External Reset Enable value from the D4100 APPS_FPGA: 1 = enabled, 0 = disabled.

6.2.24 short SetGPIO(short value, short DeviceNumber)

Sets the output values of GPIOA.2-4 to provide programmable digital outputs. Bits 4, 3, 2 of value control the output state. Bits 7, 6, 5, 1, 0 of value are not used. GPIOA.2-4 are 2.5-V CMOS outputs.

6.2.25 short GetGPIO(short DeviceNumber)

Returns the values of GPIOA.5-7 as digital inputs. Bits 7, 6, 5 of returned value are the input data. Bits 4, 3, 2, 1, 0 of returned value are not used. GPIOA.5-7 are 2.5-V CMOS inputs.

6.2.26 short GetRESETCOMPLETE(int waittime, short int DeviceNumber)

Enables global external Micromirror Clocking Pulses (resets) (see Section 6.2.23) then loop *waittime* in milliseconds for an external Micromirror Clocking Pulse (reset) trigger event to happen. If an external Micromirror Clocking Pulse (reset) occurs in the *waittime* windows, returns a 1. If *waittime* is 0, loop will run indefinitely until a Micromirror Clocking Pulse (reset) happens.

6.2.27 short GetGPIORESETCOMPLETE(short DeviceNumber)

Creates a 1 µs pulse on GPIOA.1. This can be used to trigger external hardware from the PC. Returns a 1 if successful.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		