Software Development Kit for PCIe Devices: Matlab, C++, and LabVIEW

28 June 2017

Meadowlark Optics, Inc.
5964 Iris Parkway
Frederick, CO 80530
USA

Telephone: (303) 833-4333
Fax: (303) 833-4335
sales@meadowlark.com
www.meadowlark.com

Table of Contents

1 Software Development Kit Introduction

Meadowlark Optics offers several software development kit example programs demonstrating how to interface to our SLM from Matlab, LabVIEW, and C++ with and without utilizing OverDrive Plus (ODP). Each example program demonstrates the order of operations that functions should be called in, the core functions that should be used, and how to link to our Dynamic Linked Library (DLL) to drive the Spatial Light Modulator (SLM). The purpose of this document is to outline the functions available, explain the purpose of each parameter passed, and suggest appropriate values to pass for various product options. There are two version of our DLL: Blink_SDK.dll, and Blink_SDK_C.dll. The Blink_SDK_C DLL, and corresponding Blink_SDK_C_wrapper.h file must be used for interfacing to the SLM with Matlab and LabVIEW. This document assumes familiarity with LabVIEW, Matlab, and C++ and the Meadowlark PCIe SLM hardware.

1.1 Modes of Operation

For Matlab, LabVIEW, and C++ there are two examples demonstrating use of the SLM with and without ODP enabled. Figure 1 shows the order of function calls with green indicating initialization, blue indicating the customers loop of loading images to the SLM, and red indicating the proper shut down procedure. The function calls are further detailed in Function Summary and Usage.

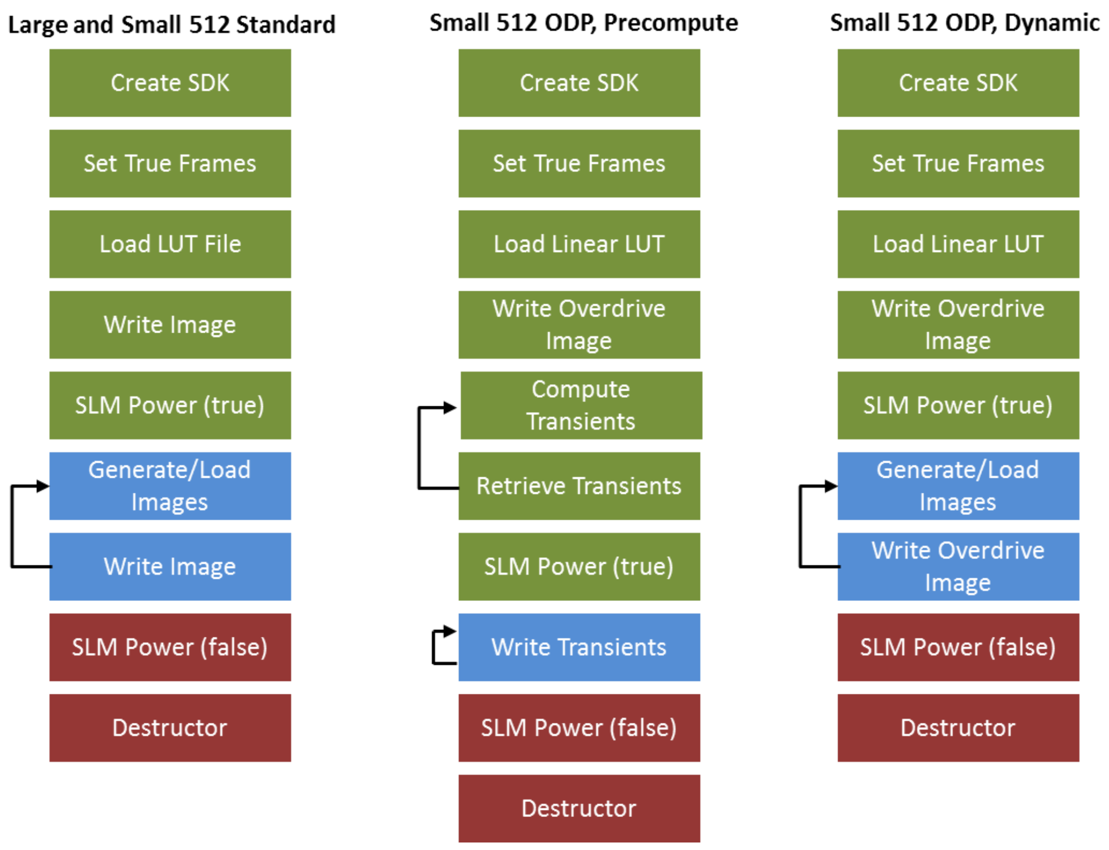
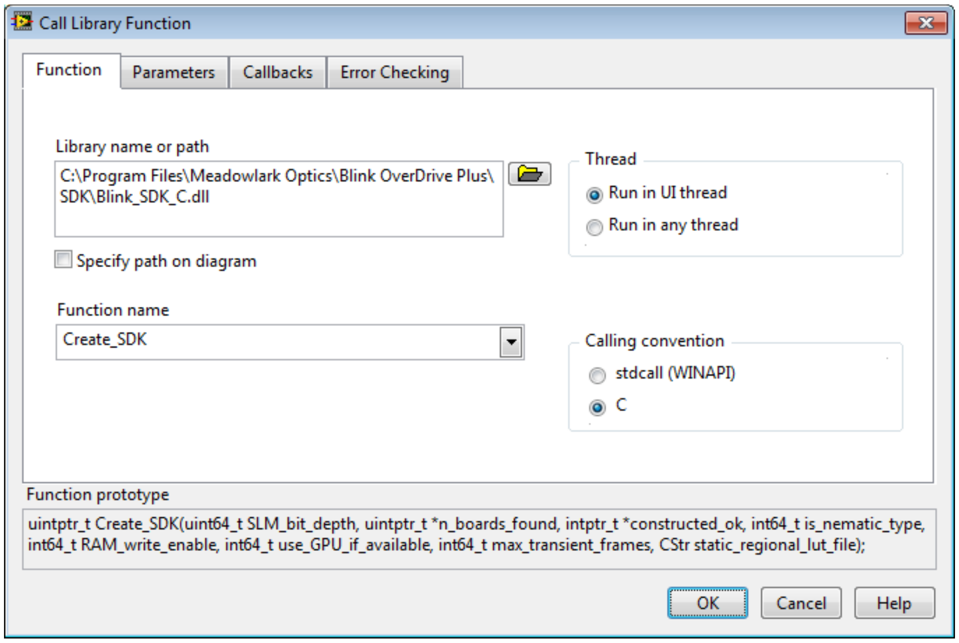


Figure 1: Function Call Order of Operations

2 Function Summary and Usage

From LabVIEW functions are accessed through Call Library Function Nodes. By double clicking on the function nodes LabVIEW allows the user to edit the function call and parameters, as shown in Fig. 2.



Library name or path: C:\Program Files\Meadowlark Optics\Blink OverDrive Plus\SDK\Blink_SDK_C.dll

Specify path on diagram: [ ]

Thread: [✓ Run in UI thread] Run in any thread

Function name: Create_SDK

Calling convention: stdcall (WINAPI) [✓ C]

Function prototype:

```
uintptr_t Create_SDK(uint64_t SLM_bit_depth, uintptr_t *n_boards_found, intptr_t *constructed_ok, int64_t is_nematic_type, int64_t RAM_write_enable, int64_t use_GPU_if_available, int64_t max_transient_frames, CStr static_regional_lut_file);
```

Figure 2: LabVIEW Call Library Function Node to interface to the Meadowlark Optics DLL

From Matlab the DLL is accessed by through the loadlibrary function, and subsequent calls to calllib as shown to follow. Loadlibrary takes two parameters: the Blink_SDK_C.dll, and the .h file. The calllib function takes 'Blink_SDK_C' as the first parameter, the function name as the second parameter, and following that the function parameters as defined in the Blink_SDK_C_wrapper.h file. Note that matlab requires a C compiler. We recommend installing Microsoft SDK 7.1, then at the matlab command prompt run mex – setup to select the installed compiler.

```
if ~libisloaded('Blink_SDK_C')
   loadlibrary('Blink_SDK_C.dll', 'Blink_SDK_C_wrapper.h');
end

sdk = calllib('Blink_SDK_C', 'Create_SDK', bit_depth, num_boards_found, constructed_okay, is_nematic_type,
RAM_write_enable, use_GPU, max_transients, lut_file);
```

2.1 Function Definitions

All function definitions listed below are based on the .h file that Matlab and LabVIEW can interface to. When interfacing from C++, the handle to the SDK is not passed (void *sdk) in each function call listed to follow.

2.1.1 *void* Create_SDK(unsigned int SLM_bit_depth, unsigned int* n_boards_found, int *constructed_ok, int is_nematic_type, int RAM_write_enable, int use_GPU_if_available, int max_transient_frames, char* static_regional_lut_file);*

Create SDK opens communication and initializes the hardware. This is a general purpose function that supports operation with or without overdrive. Parameters that are specific to Overdrive are unused. The purpose, and appropriate values of the parameters passed to this function are outlined to follow.

- SLM Bit Depth – for the large 512x512 pixel SLM the bit depth should be set to 16. For the small 512x512 with or without ODP the bit depth should be set to 8.
- Number of Boards Found – In the constructor the hardware scans the PCIe bus for Meadowlark Optics Hardware. This parameter returns the number of boards found.
- Constructed OK – This parameter tells the user if the constructor completed successfully.
- Nematic Type – This parameter is set to 1 for SLMs built with Nematic Liquid Crystal. It is set to 0 for SLMs built with Ferroelectric Liquid Crystal.
- RAM Write Enable – Utilizing RAM writes minimizes the CPU to PCIe image transfer time. This parameter should be set to 1.
- Use GPU If Available – This parameter is specific to ODP. Computation of transient frames can be computed on the GPU if a graphics card is available. This parameter should be set to 1.
- Max Transient Frames – This parameter is specific to ODP. Multiple concepts are built into ODP to improve the LC switching speed. One concept is referred to as the transient nematic effect. As shown in Fig. 1, if phase delay of a LC modulator is changed from φ_0 to φ_1, then the LC molecules relax into the new phase following approximately an exponential curve. In order to take advantage of the transient nematic effect, one instead changes the phase from φ_0 to φ_{max} to φ_1. This causes the relaxation to follow a different exponential curve until the desired phase is reached, resulting in a significantly improved switching speed. When switching between images with pixels set to arbitrary phases, the duration of time that the pixels should be set to an intermediate phase is variable based on the initial and target phase. The transient frames allow the different pixels to be held at varying voltages for an appropriate amount of time. However, on most computers each transient frame takes approximately 490 µs to load. There is a balance between minimizing the LC response time without limiting the system frame rate. In most applications we recommend this parameter be set to 10.

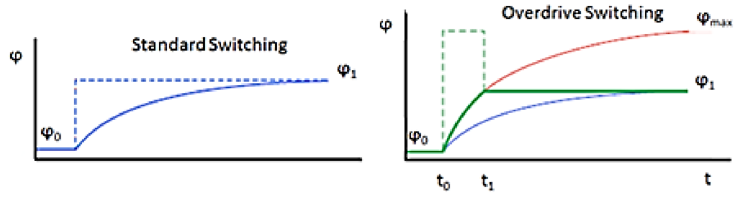


Figure 3 Standard switching, and Overdrive switching taking advantage of the transient nematic effect.

- Regional LUT – If a NULL is passed to this parameter, then ODP is disabled. Otherwise, pass the regional calibration of the LC response to applied voltage. This is a text file that was generated as a custom calibration for your SLM.

This function returns a handle that is used as a parameter to all subsequent function calls.

2.1.2 *void Set_true_frames(void *sdk, int true_frames);*

This function sets the DC balancing rate of the SLM. This function should be called immediately after the constructor. For Overdrive SLMs the true frames parameter should be set to 5. For non-overdrive operation true frames should be set to 3.

2.1.3 *int Load_LUT_file(void *sdk, int board, char* LUT_file);*

This function is used to load a calibration to the hardware that corrects for the nonlinear response of the liquid crystal to voltage. If the Look-Up Table (LUT) is correct, then the user can assume that linear increments in graylevel in user defined images translate to linear increments in phase delay on the SLM. Because images are processed through the LUT in hardware, it is important that the LUT file be loaded to the hardware prior to writing images to the SLM. The function takes a board number, which should be 1 for single SLM operation and a path to the LUT file.

- Small 512x512 and Large 512 – Use this function to load the custom LUT file that shipped with the SLM to the hardware.
- ODP – The regional calibration that was passed to Create_SDK is used instead of a global LUT file. The user should load a linear LUT to the hardware using this function, or the Load_linear_LUT function so that this feature is omitted.

2.1.4 *int Write_image(void *sdk, int board, unsigned char* image, unsigned int image_size, int wait_for_trigger, int external_pulse, unsigned int trigger_timeout_ms);*

This function writes an image to the SLM. This function takes several parameters:

- Board – This parameter tells the software which SLM the image should be written to. For single SLM operation this should be 1.
- Image – This is a pointer to a one dimensional array containing the image data. There should be 512*512 or 262,144 elements in the array, where each element is an 8-bit entry.
- Image Size – This is the height or width of the image, which should always be 512.
- Wait for Trigger – This enables external triggers to control when images are loaded to the SLM. The trigger should be a TTL pulse, applied to Input A. The hardware responds to the falling edge of the trigger. If external triggers are enabled, then the function will not return until the trigger was received. If the user attempts to trigger images before an image write is complete, then the trigger will be ignored.
- External Pulse – This provides feedback to the user notifying when images are loaded to the SLM. If using external triggers this provides acknowledgement that the external trigger was received, and a new image was loaded to the SLM. This signal is normally high, and falls low for approximately 200 ns.
- Trigger Timeout – If external triggers are enabled, but they are never received by the hardware, then the software will stop waiting for a trigger when the timeout condition is met. The units of this parameter are milliseconds. In general we recommend passing 5000 for a 5 second timeout. If the timeout condition occurs the software will post an error message notifying the user of the error, and then will load the image to the SLM that the user attempted to write.

2.1.5 *void SLM_power(void *sdk, int power_state);*

This function turns the SLM power on or off. Passing 1 will turn on the SLM power, and passing 0 will turn off the SLM power. Generally it is recommended that this function be called after a valid image is loaded to the SLM.

2.1.6 *void Delete_SDK(void *sdk);*

This function destructs the handle to the hardware, properly releases memory allocations, and shuts down the hardware. This is the last function that should be called when exiting your software.

2.2 Optional Functions

2.2.1 *const char* Get_last_error_message(void *sdk);*

This function is optional, but is a useful check of error messages.

2.2.2 *const char* Get_version_info(void *sdk);*

This function is optional, providing version information if requested by Meadowlark Optics.

2.2.3 *int Compute_TF(void *sdk, float frame_rate);*

This function is used to compute an appropriate DC balancing rate if using a Ferroelectric Liquid Crystal SLM.

2.3 ODP Functions

2.3.1 *int Is_slm_transient_constructed(void *sdk);*

A true result indicates that the overdrive frame calculation engine was properly constructed, and that its required resources are available on the system.

2.3.2 *int Load_linear_LUT(void *sdk, int board);*

When using ODP the regional calibration is used to linearize the regional response of the LC to voltage. The global calibration, which is applied in hardware should be disabled by loading a linear LUT to the hardware.

2.3.3 *int Write_overdrive_image(void *sdk, int board, unsigned char* target_phase, int wait_for_trigger, int external_pulse, unsigned int trigger_timeout_ms);*

This function loads an image to the SLM using ODP. This can be called in a loop when transients are not precomputed, or it can be called to initiate a loop of precomputed images.

- Board – This parameter tells the software which SLM the image should be written to. For single SLM operation this should be 1.

- Target Phase – This tells the software the final image that the user would like to switch to. Knowing the current image, which is blank if this is called on initialization, the software computes the transient images required to minimize the LC switching time in the transition from the current image to the target phase.
- Wait for Trigger – This enables external triggers to control when images are loaded to the SLM. The trigger should be a TTL pulse, applied to Input A. The hardware responds to the falling edge of the trigger. If external triggers are enabled, then the function will not return until the trigger was received. If the user attempts to trigger images before an image write is complete, then the trigger will be ignored.
- External Pulse – This provides feedback to the user notifying when images are loaded to the SLM. If using external triggers this provides acknowledgement that the external trigger was received, and a new image was loaded to the SLM. This signal is normally high, and falls low for approximately 200 ns.
- Trigger Timeout – If external triggers are enabled, but they are never received by the hardware, then the software will stop waiting for a trigger when the timeout condition is met. The units of this parameter are milliseconds. In general we recommend passing 5000 for a 5 second timeout. If the timeout condition occurs the software will post an error message notifying the user of the error, and then will load the image to the SLM that the user attempted to write.

*2.3.4 int Calculate_transient_frames(void *sdk, unsigned char* target_phase, unsigned int* byte_count);*

This function is used to calculate the transient images between the current image, and the target image that the customer would like to transition to. This function is only used if you are pre-computing the transient images. If the customer is dynamically computing the transients, then Write_overdrive_image should be used.

- Target Phase – This tells the software the final image that the user would like to switch to. Knowing the current image, which is blank if this is called on initialization, the software computes the transient images required to minimize the LC switching time in the transition from the current image to the target phase.
- Byte Count – This returns the size of the transient buffer required to store the transient frames

*2.3.5 int Retrieve_transient_frames(void *sdk, unsigned char* frame_buffer);*

This function is used to return the transient frames to the user so that the user can sequence through the pre-computed transient data in a loop. The user passes in a buffer to be filled, that was allocated using the byte_count retuned by Calculate_transient_frames.

*2.3.6 int Write_transient_frames(void *sdk, int board, unsigned char* frame_buffer, int wait_for_trigger, int external_puls, unsigned int trigger_timeout_ms);*

This function is used to write transient frames to the SLM. Note that the transient frames hold the target phase defined by the user Calculate_transient_frames as the last frame in the image sequence.

- Frame Buffer – This is the buffer of transient images plus the target phase as computed by Calculate_transient_frames.
- Wait for Trigger – This enables external triggers to control when images are loaded to the SLM. The trigger should be a TTL pulse, applied to Input A. The hardware responds to the falling edge of the trigger. If

external triggers are enabled, then the function will not return until the trigger was received. If the user attempts to trigger images before an image write is complete, then the trigger will be ignored.
- External Pulse – This provides feedback to the user notifying when images are loaded to the SLM. If using external triggers this provides acknowledgement that the external trigger was received, and a new image was loaded to the SLM. This signal is normally high, and falls low for approximately 200 ns.
- Trigger Timeout – If external triggers are enabled, but they are never received by the hardware, then the software will stop waiting for a trigger when the timeout condition is met. The units of this parameter are milliseconds. In general we recommend passing 5000 for a 5 second timeout. If the timeout condition occurs the software will post an error message notifying the user of the error, and then will load the image to the SLM that the user attempted to write.

2.4 Optional ODP Functions

*2.4.1 int Load_overdrive_LUT_file(void *sdk, char* static_regional_lut_file);*

The regional calibration linearizes the LC response to applied voltage. This is a text file that was generated as a custom calibration for your SLM. If the file you intend to use changes from that passed to the constructor, then this function can be used to re-load the regional calibration.

*2.4.2 int Stop_sequence (void *sdk);*

If using external triggers, and an image write has been initiated but an external trigger is not received, then the user can abort the image write using this function. In order to utilize this function it is necessary to have a multi-threaded application.

Please feel free to contact us with any questions you may have.

For questions regarding customer support for Meadowlark Optics products, please contact us by telephone or by e-mailing support@meadowlark.com.

For questions regarding purchasing and pricing of additional Meadowlark Optics products, please contact us by telephone or by e-mailing sales@meadowlark.com.