API for controlling DMD

Table of Contents

Load libraries in MATLAB *(api_library)* 1
Connect/ Allocate a DMD device *(api_allocate)* 2
Inquire device/ controller parameters *(api_inquire)* 2
Reset the DMD device *(api_reset)* 4
Clear the DMD mirrors *(api_clear)* 5
Load image on the mirrors *(api_load)* 6
Free the DMD Device after use *(api_free)* 6
Return: Success / error reporting *(return_check)* 6

This API controls Digital Micromirror Devices (DMD) device controlled by ViALUX ALP controller. The library is provided by ViALUX GmbH, which is loaded into the environment. Since the library is in C++, there needs to be a working C++ compiler installed and setup with MATLAB. Compatible compilers could be found at http://www.mathworks.com/support/compilers/R2014b/index.html (careful about the version).

Necessary documentation and libraries can be found in accompanying media (CDs, Drives etc,) from ViALUX. Read about the original VIALUX API in documentation.

- This API was developed with grateful aid from Dr. Martin Vogel (Max Planck Institute of Biophysics) and VIALUX. This is a simpler subset of the ALP Tool provided by Dr. Vogel. (http://www.mathworks.com/matlabcentral/fileexchange/46673-alptool)
- Get ALP drivers at (http://www.vialux.de/transfer/alp-4.2/ALP42_install.exe OR http://www.vialux.de/transfer/alp-4.1/ALP41_install.exe)
- Careful about the versions of library you load - x64/ x86. This API is for basic - not high speed.
- Comments/ correspondence should be addressed to **nakulbende [at] gmail [dot] com**
- Following functions can be used. An exmaple code with a typical routine is included as **api_control.m**. This code contains all the functions to be used, in the right order to load library, connect the mirrors and load images.

Load libraries in MATLAB *(api_library)*

Loads a shared library and it's functions in MATLAB environment.

INPUTS: (both without file extensions)

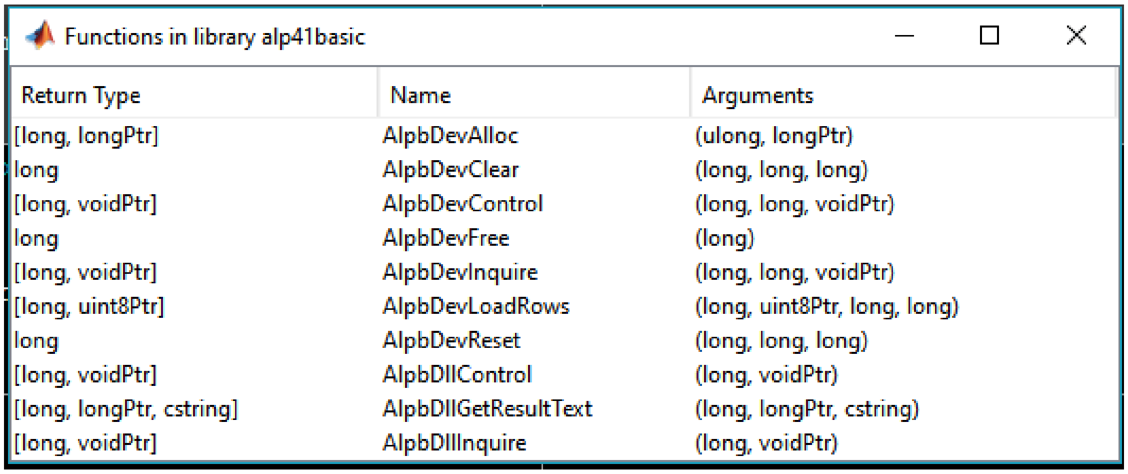
dll_name = name of the dll file

dll_header = name of the header file

OUTPUTS:

return_lib = 'Library is loaded'; or 'Error: Library was not loaded'

(Opens a list of functions available in library in a seperate window)



Functions in library alp41basic

Return Type	Name	Arguments
[long, longPtr]	AlpbDevAlloc	(ulong, longPtr)
long	AlpbDevClear	(long, long, long)
[long, voidPtr]	AlpbDevControl	(long, long, voidPtr)
long	AlpbDevFree	(long)
[long, voidPtr]	AlpbDevInquire	(long, long, voidPtr)
[long, uint8Ptr]	AlpbDevLoadRows	(long, uint8Ptr, long, long)
long	AlpbDevReset	(long, long, long)
[long, voidPtr]	AlpbDllControl	(long, voidPtr)
[long, longPtr, cstring]	AlpbDllGetResultText	(long, longPtr, cstring)
[long, voidPtr]	AlpbDllInquire	(long, voidPtr)

Figure. 1 Loaded library and available functions

```
function [return_lib] = api_library(dll_name, dll_header)
```

Connect/ Allocate a DMD device *(api_allocate)*

Connects the device with MATLAB, and generates a device handle which will be used as an address for subsequent operations

INPUTS: (both without file extensions)

dll_name = name of the dll file

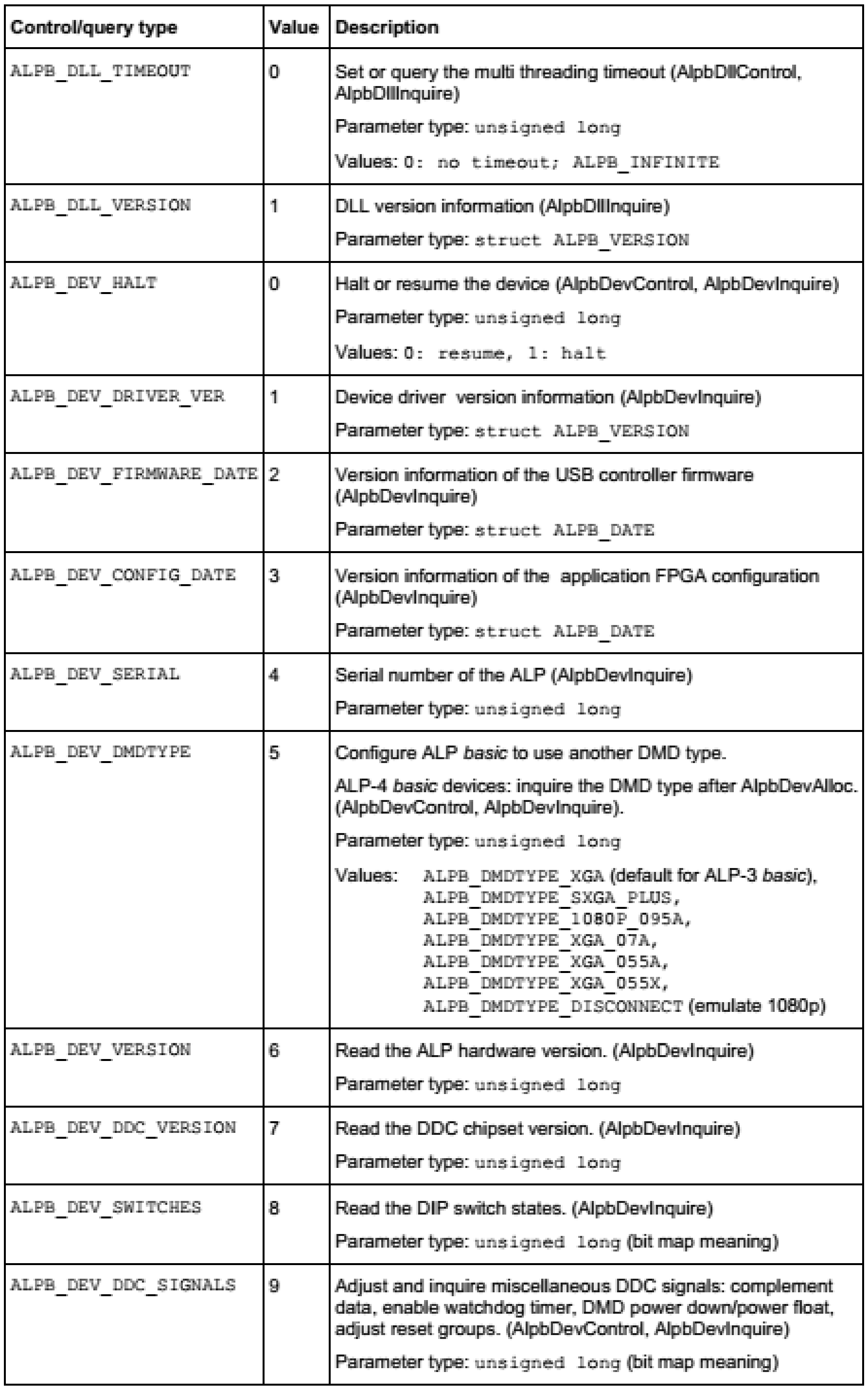
OUTPUT:

hdevice = device handle generated by allocate function

```
function [return_allocate, hdevice] = api_allocate(dll_name)
```

Inquire device/ controller parameters *(api_inquire)*

Sends a query to the device, and stores the value in an out pointer. The query types can be (found in documentation):



Control/query type	Value	Description
ALPB_DLL_TIMEOUT	0	Set or query the multi threading timeout (AlpbDllControl, AlpbDllInquire) Parameter type: `unsigned long` Values: `0: no timeout; ALPB_INFINITE`
ALPB_DLL_VERSION	1	DLL version information (AlpbDllInquire) Parameter type: `struct ALPB_VERSION`
ALPB_DEV_HALT	0	Halt or resume the device (AlpbDevControl, AlpbDevInquire) Parameter type: `unsigned long` Values: `0: resume, 1: halt`
ALPB_DEV_DRIVER_VER	1	Device driver version information (AlpbDevInquire) Parameter type: `struct ALPB_VERSION`
ALPB_DEV_FIRMWARE_DATE	2	Version information of the USB controller firmware (AlpbDevInquire) Parameter type: `struct ALPB_DATE`
ALPB_DEV_CONFIG_DATE	3	Version information of the application FPGA configuration (AlpbDevInquire) Parameter type: `struct ALPB_DATE`
ALPB_DEV_SERIAL	4	Serial number of the ALP (AlpbDevInquire) Parameter type: `unsigned long`
ALPB_DEV_DMDTYPE	5	Configure ALP *basic* to use another DMD type. ALP-4 *basic* devices: inquire the DMD type after AlpbDevAlloc. (AlpbDevControl, AlpbDevInquire). Parameter type: `unsigned long` Values: `ALPB_DMDTYPE_XGA` (default for ALP-3 *basic*), `ALPB_DMDTYPE_SXGA_PLUS`, `ALPB_DMDTYPE_1080P_095A`, `ALPB_DMDTYPE_XGA_07A`, `ALPB_DMDTYPE_XGA_055A`, `ALPB_DMDTYPE_XGA_055X`, `ALPB_DMDTYPE_DISCONNECT` (emulate 1080p)
ALPB_DEV_VERSION	6	Read the ALP hardware version. (AlpbDevInquire) Parameter type: `unsigned long`
ALPB_DEV_DDC_VERSION	7	Read the DDC chipset version. (AlpbDevInquire) Parameter type: `unsigned long`
ALPB_DEV_SWITCHES	8	Read the DIP switch states. (AlpbDevInquire) Parameter type: `unsigned long` (bit map meaning)
ALPB_DEV_DDC_SIGNALS	9	Adjust and inquire miscellaneous DDC signals: complement data, enable watchdog timer, DMD power down/power float, adjust reset groups. (AlpbDevControl, AlpbDevInquire) Parameter type: `unsigned long` (bit map meaning)

Table.1 Queries to be used in alp_inquiry, alp_control

INPUTS:

dll_name = Loaded control library *hdevice* = device handle generated by allocate function

query = Query type: Pg 6, returns in Pg. 7, Pg. 16, Pg. 17

Common query commands (Pg. 16), non-exhaustive, please refer to documentation

- 0: Timeout
- 1: dll Version, driver version
- 2. Halt, firmware date
- 3. Configuration date
- 4. Device serial
- 5. DMD Type (return values in Pg 7)
- 6. Hardware version
- 7. Chipset version
- 8. DIP switches
- 9. DDC Signals

OUTPUT:

return_queryptr = C style pointer with the readout from device/ controlled about the specific query. Data type depends on the query, and can be found out by using the command get(*return_queryptr*)

```
function [return_inquiry, return_query] = api_inquire(dll_name, hdevice, query)
```

Reset the DMD device *(api_reset)*

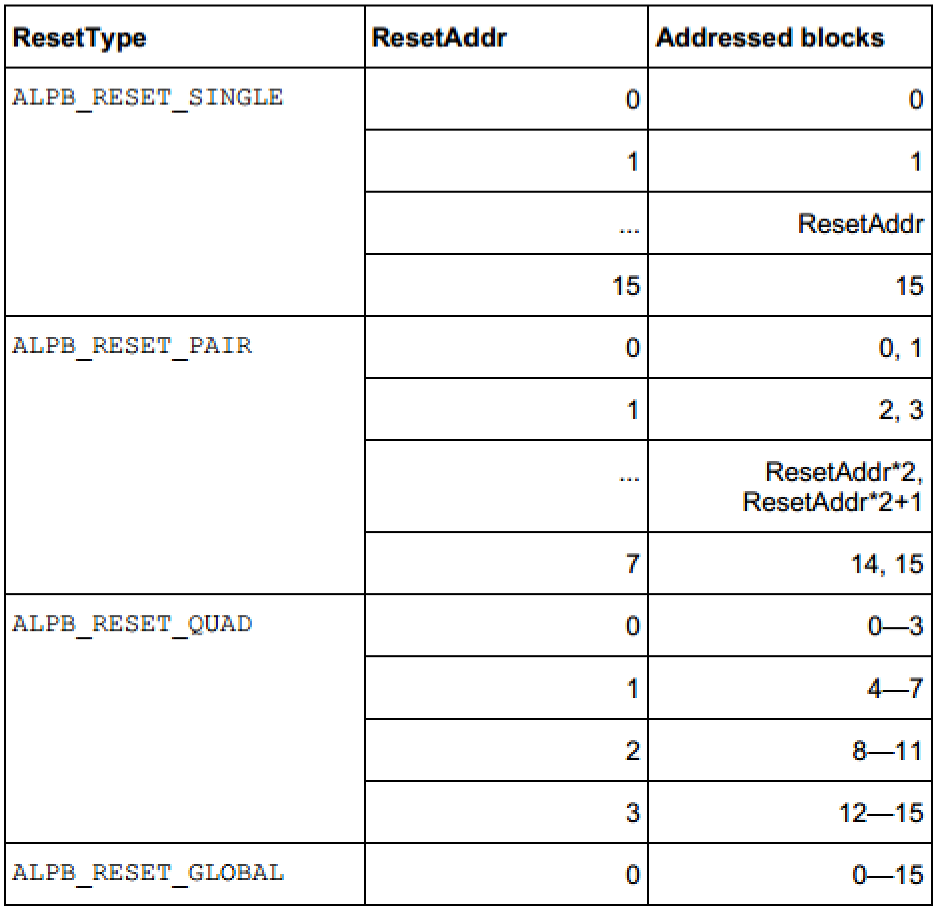
Reset the DMD device to load another image. Should be performed before any clear function. The reset operation itself takes the same time to finish, independent of how many mirrors are affected.

INPUTS:

dll_name = Loaded control library *hdevice* = device handle generated by allocate function *reset_mode* = first block to be cleared Pg. 22 of API guide,

- 1 : Single
- 2 : Pair
- 3 : Quad
- 4 : Global

reset_address = address of block to be reset (0 for global), see guide for others



ResetType	ResetAddr	Addressed blocks
ALPB_RESET_SINGLE	0	0
	1	1
	...	ResetAddr
	15	15
ALPB_RESET_PAIR	0	0, 1
	1	2, 3
	...	ResetAddr*2, ResetAddr*2+1
	7	14, 15
ALPB_RESET_QUAD	0	0—3
	1	4—7
	2	8—11
	3	12—15
ALPB_RESET_GLOBAL	0	0—15

Table.2 Reset block assignment for different options

OUTPUT:

return_reset = Return for success/ error reporting

```
function [return_reset] = api_reset(dll_name, hdevice, reset_mode, reset_address)
```

Clear the DMD mirrors *(api_clear)*

The clear operation sets the memory content of whole reset blocks to logic ’0’

INPUTS:

dll_name = Loaded control library

hdevice = device handle generated by allocate function

first_block = first block to be cleared (0)

last_block = last block to be cleared (15)

OUTPUT:

return_clear = Return for success/ error reporting

```
function [return_clear] = api_clear(dll_name, hdevice, first_block, last_block)
```

Load image on the mirrors *(api_load)*

Send an image to the mirrors, and display it. Careful about the lags, if using this in a loop. This should always be performaed in following to Reset > Clear command.

INPUTS:

dll_name = Loaded control library *hdevice* = device handle generated by allocate function

image = image matrix should be in 0/1. Dimensions 768X1024 (rowsXcolumns). Note that C style structures are transpose equivalant of the MATLAB counterparts.

first_row = first row to be loaded (0)

last_row = last row to be loaded (767)

OUTPUT:

return_load = Return for success/ error reporting

```
function [return_load] = api_load(dll_name, hdevice, image, first_row, last_row)
```

Free the DMD Device after use *(api_free)*

Frees the device, and returns the mirrors to a floating position. !!Alwyas perform this beofre shutting off the mirrors!!

INPUTS:

dll_name = Loaded control library

hdevice = device handle generated by allocate function

OUTPUT:

return_free = Return for success/ error reporting

```
function [return_free] = api_free(dll_name, hdevice)
```

Return: Success / error reporting *(return_check)*

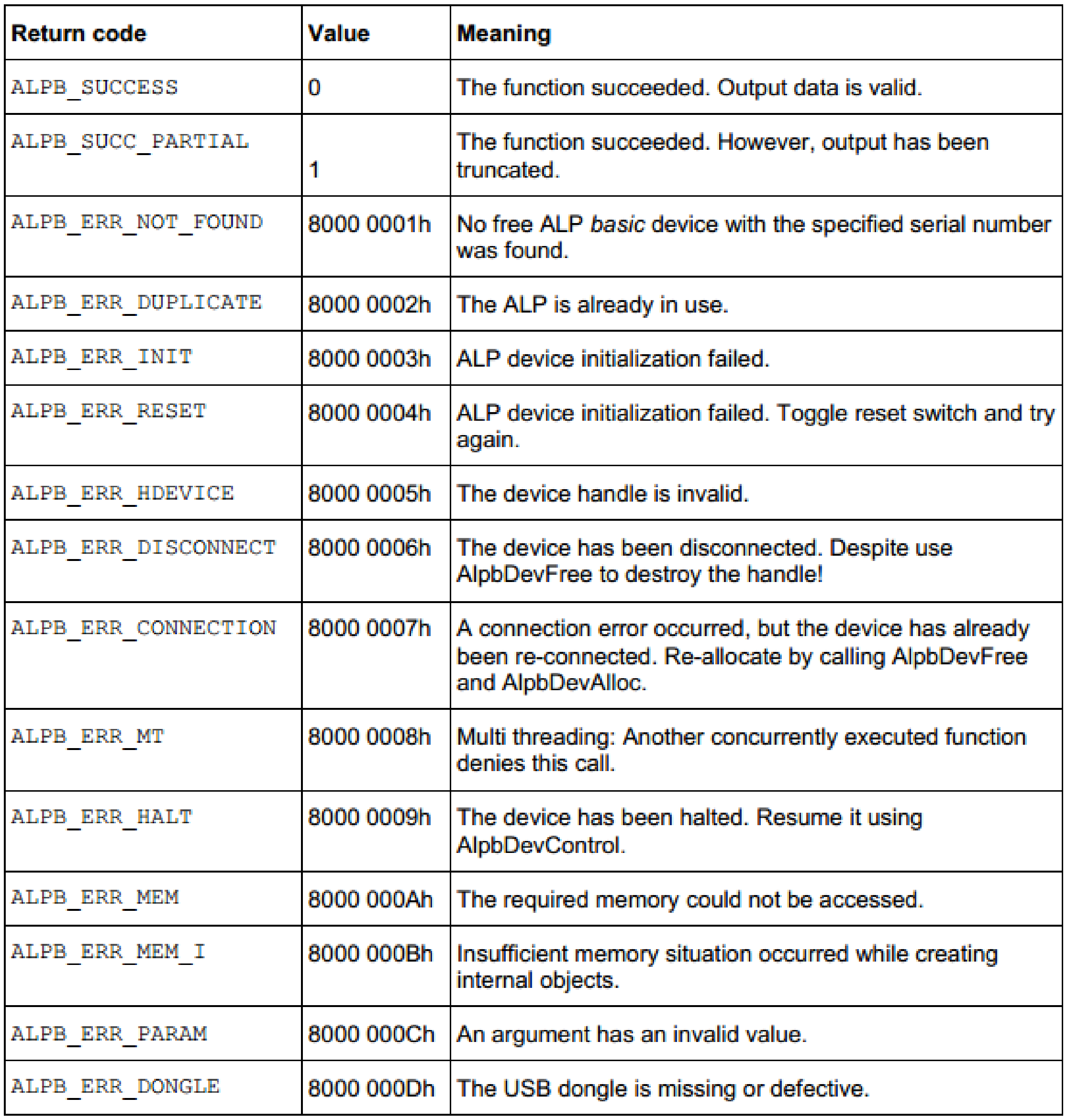
Check the return from other functions - Check documentation (Pg. 9)

INPUTS:

return_value = Return for success/ error reporting

OUTPUT:

out_signal = Return for success/ error reporting



Return code	Value	Meaning
ALPB_SUCCESS	0	The function succeeded. Output data is valid.
ALPB_SUCC_PARTIAL	1	The function succeeded. However, output has been truncated.
ALPB_ERR_NOT_FOUND	8000 0001h	No free ALP *basic* device with the specified serial number was found.
ALPB_ERR_DUPLICATE	8000 0002h	The ALP is already in use.
ALPB_ERR_INIT	8000 0003h	ALP device initialization failed.
ALPB_ERR_RESET	8000 0004h	ALP device initialization failed. Toggle reset switch and try again.
ALPB_ERR_HDEVICE	8000 0005h	The device handle is invalid.
ALPB_ERR_DISCONNECT	8000 0006h	The device has been disconnected. Despite use AlpbDevFree to destroy the handle!
ALPB_ERR_CONNECTION	8000 0007h	A connection error occurred, but the device has already been re-connected. Re-allocate by calling AlpbDevFree and AlpbDevAlloc.
ALPB_ERR_MT	8000 0008h	Multi threading: Another concurrently executed function denies this call.
ALPB_ERR_HALT	8000 0009h	The device has been halted. Resume it using AlpbDevControl.
ALPB_ERR_MEM	8000 000Ah	The required memory could not be accessed.
ALPB_ERR_MEM_I	8000 000Bh	Insufficient memory situation occurred while creating internal objects.
ALPB_ERR_PARAM	8000 000Ch	An argument has an invalid value.
ALPB_ERR_DONGLE	8000 000Dh	The USB dongle is missing or defective.

Table.3 Return codes

```
function [out_signal] = return_check(return_value);
```

Published with MATLAB® R2014a