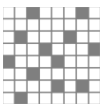
ALP *basic*

Application Programming Interface

VIALUX



ViALUX Messtechnik + Bildverarbeitung GmbH

Am Erlenwald 10

09128 Chemnitz

Germany

Phone: +49 (0) 371 33 42 47 0

Fax: +49 (0) 371 33 42 47 10

Web: www.vialux.de

E-Mail: dlp@vialux.de

© 2006-2020 ViALUX GmbH. All rights reserved.

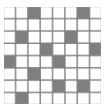


Table of Contents

1 General remarks 1
2 Naming conventions 2
3 Document version history 3
4 Constants 5
4.1 Return values 5
4.2 Control and query types 6
4.3 Others 7
5 Data types 8
5.1 C-style declarations 8
5.2 Memory organization 8
6 Functions 9
6.1 AlpbDllControl 9
6.2 AlpbDllInquire 10
6.3 AlpbDllGetResultText 11
6.4 AlpbDevAlloc 12
6.5 AlpbDevControl 13
6.6 AlpbDevInquire 16
6.7 AlpbDevFree 17
6.8 AlpbDevLoadRows 18
6.9 AlpbDevClear 20
6.10 AlpbDevReset 20
7 Hints 22
7.1 DMD type support 22
7.2 ALP Serial Number and Labeling 23
7.3 DMD Float operation 23

Tables

Table 1: Document version history 4
Table 2: Return codes 5
Table 3: Control types 7
Table 4: Other constants 7
Table 5: Data types 8
Table 6: Control types (AlpbDllControl) 9
Table 7: Query types (AlpbDllInquire) 10
Table 8: Control types (AlpbDevControl) 14
Table 9: Query types (AlpbDevInquire) 17
Table 10: Pixel format 19
Table 11: Image data structure 19
Table 12: Reset block assignment by (ResetType, ResetAddr) 21

1 General remarks

ALP *basic* is a controller suite compatible with various versions of DLP® Discovery™ boards. ALP-3 *basic* supports the DLP® Discovery™ 3000 Starter Kit Board. ALP-4 *basic* supports DLP® Discovery™ 4100 based hardware (the DLP® Discovery™ 4100 Developer Kit). ALP-4.2 is the according software for ViALUX V-7000 Modules (V4100 Boards).

The ALP *basic* suite comes with all components required for the use of the DLP® Discovery™ Boards including a software DLL. The following notes describe general software organization rules applicable to the whole library.

The image display on a DMD is divided into two different operations. Data operations load image data in on-chip SRAM memory cells. In a subsequent reset operation the mirrors are flipped to +12° or -12° positions, respectively, according to the SRAM data.

There are two types of data operations. Loading data (*AlpbDevLoadRows*) transfers image data row by row to the DMD. The clear operation (*AlpbDevClear*) sets the memory content of whole reset blocks to logic ’0’.

A Reset operation (*AlpbDevReset*) addresses the micro mirrors in terms of reset blocks. Single blocks, groups of two or four blocks, or even all mirrors (global reset) can be reset at the same time. The reset operation itself takes the same time to finish, independent of how many mirrors are affected.

ALP-3, ALP-4.1, and ALP-4.2 *basic* APIs reside in different DLL files: alp3basic.dll, alp41basic.dll, alpV42basic.dll.

- Exported function names have device-dependent prefixes. ALP-3 *basic* (alp3basic.dll) uses names like “Alp3b*”. ALP-4 *basic* (alp41basic.dll, alpV42basic.dll) have “Alpb*”. This document only uses the Alpb* names. Please substitute it by Alp3b when using ALP-3 *basic* devices.
- There are two configurations of the ALP *basic* DLL having different names. Alp*basic.dll uses the “cdecl” calling convention. The library alp*basicS.dll exports functions using the “stdcall” calling convention and redirects them to the cdecl DLL. This might be required for development environments that do not support the C calling convention. Function names and argument lists are equal in both configurations.
- For C and C++ users a header (alpbasic.h) and a static library (alp3basic.lib, alp41basic.lib, or alpV42basic.lib) file is provided together with the DLL.
- ALP *basic* devices are identified in function calls by handles of type ALPB_HDEVICE. This handle is generated by AlpbDevAlloc.

- After usage the device handle must be freed using the AlpbDevFree function. This function requires that no other function AlpbDevLoadRows, AlpbDevClear, or AlpbDevReset is running (applicable in multi threading programs).
- These functions can be canceled using AlpbDevControl, type ALPB_DEV_HALT.
- Return values are 32 bit wide. The most significant bit denotes success (0) or error (1). This allows for general error handling:
  ```
  if (ReturnValue >> 31) { /* general error handling */ }
  ```
  The other bits carry detailled information about the kind of error and maybe which way the function succeeded:
  ```
  if (ALPB_ERR_HDEVICE == ReturnValue) { /* device handle invalid */
  }
  ```
- Function output is always written to user provided memory. Therefore sometimes pointer arguments are required.
  The return value tells only the outcome of the function. One error code has the same meaning for all functions.
- Pointer arguments are marked in this document as “In”, “Out”, or “InOut”. This means that the memory is read, written, or both respectively by the called function.

2 Naming conventions

- Constants: upper-case, prefix “ALPB_”
- Return values (subset of “Constants”): prefix “ALPB_ERR_”, “ALPB_SUCC_”
- Data types and structs: prefix “ALPB_”
- Function names are concatenated of three parts:
  prefix (“Alpb”), referred object type (“Dll” or “Dev”), function (“Control”, “Inquire”, etc.)

3 Document version history

Version	Description	Date
1	Initial release	2006-07-07
2	Minor changes	2006-08-11
3	Minor changes	2006-10-20
4	Applies to alp3basic.dll ≥ 1.0.0.7 • Clarify data and reset operation. • Introduce DMD type support and add SXGA+ DMD format. Affected: Table 3, Table 4, Table 11, 6.8 AlpbDevLoadRows, 6.10 AlpbDevReset, 7.1 DMD type support • Change power down precautions (PWR_FLOAT). Affected: 6.7 AlpbDevFree, 7.3 DMD Float operation • Remove Alp3bDevFloat function. • Remove table “DMD formats”. This information is now available in the document “ALP-3 *basic* Supplement”	2007-11-22
5	Changes due to introduction of ALP-4 *basic*. Applies to alp4basic.dll ≥ 1.0.0.1 • Revise section 1 General remarks • Rename all constants from “ALP3B_*” to “ALPB_*” Their values are compatible, so the new names can be used with former ALP-3 *basic* API versions. • Rename functions from “Alp3b*” to “Alpb*” (applies to ALP-4 *basic*) • Add return code `ALPB_ERR_DONGLE`: Table 2 and most of the AlpbDev* functions. • Add new DMD types `ALPB_DMDTYPE_1080P_095A`, `ALPB_DMDTYPE_XGA_07A`, `ALPB_DMDTYPE_XGA_055A`, `ALPB_DMDTYPE_XGA_055X` (Table 3, Table 4, Table 8, Table 11) • Add reset type `ALPB_RESET_QUAD` (Table 4, Table 12) • Add control and query types `ALPB_DEV_VERSION`, `ALPB_DEV_DDC_VERSION`, `ALPB_DEV_SWITCHES`, `ALPB_DEV_DDC_SIGNALS` (Table 3, Table 8, Table 9) • Add explanation of `ALPB_DEV_DDC_SIGNALS` in section 0 • AlpbDevControl • Add section 7.2 ALP Serial Number and Labeling • Revise section 7.3 DMD Float operation	2008-10-15
6	Introduction of ALP-4.1 basic. Applies to alp41basic.dll ≥ 1.0.0.1 • Revise section 1 General remarks • Add new DMD type `ALPB_DMDTYPE_DISCONNECT` (Table 3, Table 4, section 7.1 DMD type support) • Device version (Table 9) • Section 7.2 ALP Serial Number and Labeling	2009-09-23

Version	Description	Date
7	Introduction of ALP-4.2 *basic*. Applies to alpV42basic.dll ≥ 1.0.0.1 • Revise section 1 General remarks • Section ALPB_DEV_DDC_SIGNALS, Bit 2: DMD Power Down / Mirror Float (PWR_FLOAT) • Device version (Table 9) • Section 7.2 ALP Serial Number and Labeling	2010-06-21
8	• Remove obsolete hardware version ALP-4.0. • Review Section 7.2 ALP Serial Number and Labeling • Specify WUXGA DMD format (ALP-4.1)	2013-05-30
9	• Added WXGA DMD	2020-03-23

Table 1: Document version history

4 Constants

4.1 Return values

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 2: Return codes

4.2 Control and query types

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_WUXGA_096A, ALPB_DMDTYPE_WXGA_S450, 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)

Control/query type	Value	Description
`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 3: Control types

4.3 Others

Code	Value	Usage (see also)
`ALPB_RESET_SINGLE`	0	Reset type (function AlpbDevReset)
`ALPB_RESET_PAIR`	1	
`ALPB_RESET_QUAD`	2	
`ALPB_RESET_GLOBAL`	4	
`ALPB_INFINITE`	FFFF FFFFh	infinite multi threading wait timeout (control type ALPB_DLL_TIMEOUT, function AlpbDllControl)
`ALPB_DMDTYPE_XGA`	1	Parameter for AlpbDevControl, AlpbDevInquire when ControlType=ALPB_DEV_DMDTYPE (see also Table 3)
`ALPB_DMDTYPE_SXGA_PLUS`	2	
`ALPB_DMDTYPE_1080P_095A`	3	
`ALPB_DMDTYPE_XGA_07A`	4	
`ALPB_DMDTYPE_XGA_055A`	5	
`ALPB_DMDTYPE_XGA_055X`	6	
`ALPB_DMDTYPE_WUXGA_096A`	7	
`ALPB_DMDTYPE_WXGA_S450`	8	
`ALPB_DMDTYPE_DISCONNECT` (ALP-4 *basic* emulates a 1080p DMD in this case)	255	

Table 4: Other constants

5 Data types

5.1 C-style declarations

```
typedef long ALPB_HDEVICE;

struct ALPB_VERSION {
       short        Version1;
       short        Version2;
       short        Version3;
       short        Build;
};

struct ALPB_DATE {
       short        Year;              // year AD
       short        Month;             // month: 1..12
       short        Day;               // day of month: 1..31
};
```

5.2 Memory organization

If another programming language than C/C++ is used please refer to the following table to create your own data structures for ALP *basic* API calls.

Type (size in bytes)	Member data type (size)	Member (byte offset)
`ALPB_HDEVICE` (4)	`long` (4)	Device handle (0)
`ALPB_VERSION` (8)	`short` (2)	Version1 (0)
	`short` (2)	Version2 (2)
	`short` (2)	Version3 (4)
	`short` (2)	Build (6)
`ALPB_DATE`(6)	`short` (2)	Year AD (0)
	`short` (2)	Month 1..12 (2)
	`short` (2)	Day 1..31 (4)

Table 5: Data types

6 Functions

6.1 AlpbDllControl

Syntax

```
long AlpbDllControl(long ControlType, void *pValue)
```

Description

Set up global behaviour of the DLL in the current process. ControlType specifies which setting is to be adjusted. Because the data types of settings can differ, the value (*pValue) is provided as the data addressed by the void pointer pValue.

Parameters

ControlType	Selection of the setting to adjust
pValue	In pointer to the new value

ControlType	Data type (bytes)	Description
`ALPB_DLL_TIMEOUT`	`unsigned long` (4)	Some functions execution is mutual exclusive in multi threading usage. If one function is called while another is currently active, it can • wait an indeterminate time for the other function to finish (*pValue=ALPB_INFINITE) • immediately return error code ALPB_ERR_MT (*pValue=0) The setting only influences subsequent function calls. Default value: ALPB_INFINITE

Table 6: Control types (AlpbDllControl)

Return codes

ALPB_SUCCESS

ALPB_ERR_MT

ALPB_ERR_MEM

ALPB_ERR_PARAM

6.2 AlpbDllInquire

Syntax

```
long AlpbDllInquire(long QueryType, void *pValue)
```

Description

Query settings global in the DLL and the current process. ControlType acts as a selector whilst pValue points to a variable of the according data type.

Valid query types are a superset of the control types used in the function AlpbDllControl.

Parameters

QueryType	Selection of the data to be queried
pValue	Out pointer to a memory range where output is written to

QueryType	Data type (bytes)	Description
`ALPB_DLL_TIMEOUT`	`unsigned long` (4)	see function AlpbDllControl
`ALPB_DLL_VERSION`	`ALPB_VERSION` (8)	Query version information. This information is useful for comparision against the release notes of the API.

Table 7: Query types (AlpbDllInquire)

Return codes

ALPB_SUCCESS
ALPB_ERR_MT
ALPB_ERR_MEM
ALPB_ERR_PARAM

6.3 AlpbDllGetResultText

Syntax

```
long AlpbDllGetResultText(long RetVal, long *pSize, char *pStr)
```

Description

Retrieve a text message for the return value.

*pSize is interpreted as the number of bytes already allocated for the string. The function outputs the number of bytes necessary for this string in the same parameter *pSize.

If *pSize>0, a memory area of size bytes must be allocated first, pointed to by *pStr. This function writes an error message to this memory area and appends a 0 (NULL, zero) character.

If *pSize is less than the message needs, including the trailing NULL, it is truncated after size-1 characters. In this case, ALPB_SUCC_PARTIAL is returned.

The NULL character can not be written, if *pSize=0. However, the necessary size is put out, and ALPB_SUCC_PARTIAL is returned.

Parameters

RetVal	Return value. Result of another Alpb function.
*pSize	InOut pointer. In: Number of bytes allocated for the message string. Out: Number of bytes necessary for the message string.
*pStr	Out pointer to a memory area to receive the error message text.

Return codes

ALPB_SUCCESS
ALPB_SUCC_PARTIAL
ALPB_ERR_MEM
ALPB_ERR_PARAM

6.4 AlpbDevAlloc

Syntax

```
long AlpbDevAlloc(unsigned long DeviceNum, ALPB_HDEVICE *hDevice)
```

Description

Initialize the ALP *basic* device. The returned handle *hDevice is used to identify this device in subsequent function calls. If non-zero the device with the serial number specified by DeviceNum is allocated. If DeviceNum is 0 (zero) then the first free device is used. Serial numbers can be queried using the AlpbDevInquire function.

Release the device after usage using the function AlpbDevFree.

Parameters

DeviceNum	Serial number of the device to allocate or 0 (zero)
hDevice	Out pointer to an ALP device handle variable

Return codes

ALPB_SUCCESS
ALPB_ERR_NOT_FOUND
ALPB_ERR_DUPLICATE
ALPB_ERR_INIT
ALPB_ERR_RESET
ALPB_ERR_MT
ALPB_ERR_MEM
ALPB_ERR_MEM_I
ALPB_ERR_DONGLE

6.5 AlpbDevControl

Syntax

```
long AlpbDevControl(ALPB_HDEVICE hDevice, long ControlType,
                    void *pValue)
```

Description

Change device settings. ControlType selects what to adjust. Because the data can have different types, the value (*pValue) is provided as the data addressed by the void pointer pValue.

Parameters

hDevice	ALP *basic* device handle (retrieve it using the function AlpbDevAlloc)
ControlType	Selection of the setting to adjust
pValue	In pointer to the new value

ControlType	Data type (bytes)	Description
`ALPB_DEV_HALT`	`unsigned long` (4)	This flag provides the possibility to terminate running calls of AlpbDevLoadRows, AlpbDevClear, and AlpbDevReset. Future calls to these functions are forbidden, too. This may be helpful, because AlpbDevFree requires that none of these functions is currently executed. *pValue=1: halt the device *pValue=0: resume (allow future calls)

`ALPB_DEV_DMDTYPE`	`unsigned long` (4)	Configure ALP *basic* to use the selected DMD type
		Note for ALP-3: There is no means to retrieve information about which type of DMD is connected. Therefore the user has to set up the API according to the actually connected DMD.
		Note for ALP-4: After allocation the API is automatically set to the connected DMD type.
		`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_WUXGA_096A`
		`ALPB_DMDTYPE_WXGA_S450`

ControlType	Data type (bytes)	Description
`ALPB_DEV_DDC_SIGNALS`	`unsigned long` (4)	There are additional DDC control signals. They influence device operation and are made accessible to user programs through the ALP *basic* API using this control type. Please see the notes below this table for details. Bit 0: complement image data when '1' Bit 1: disable watchdog timer when '1' Bit 2: float all DMD mirrors for power down operation when '1' Bit 3: adjust reset groups Other bits are currently not supported and must be '0'.

Table 8: Control types (AlpbDevControl)

ALPB_DEV_DDC_SIGNALS, Bit 0: Complement Data (COMP_DATA)
This bit controls the DDC signal COMP_DATA. When set to '1', the DMD internally complements its data inputs prior to loading the data into the mirror array.

Note: As stated in the DDC data sheet, this control cannot be expected to take effect immediately upon assertion. So it is recommended to set it to one value and to not adjust it during normal system operation.

ALPB_DEV_DDC_SIGNALS, Bit 1: Disable Watchdog Timer (WDT_ENABLEZ)
Texas Instruments Inc. recommends that the time between mirror resets on any given DMD mirror block does not exceed 10 seconds. A watchdog timer in the DDC initiates a global DMD mirror reset when there has not been any DMD mirror block reset within the last 10 seconds.

The watchdog timer can be disabled by setting this control bit to '1'. In this case the user must take care for periodical mirror resets for himself.

Note: While leaving the mirrors landed for extended periods of time causes no known damage, there have been no extensive lifetime studies under this condition.

ALPB_DEV_DDC_SIGNALS, Bit 2: DMD Power Down / Mirror Float (PWR_FLOAT)
When the DMD is not in use, it can be disabled. In power-down mode all mirrors are in the flat state (about 0°).

Power-down mode is entered when

- this control bit is set,
- the push button is pressed (ALP-3 and ALP-4.0: SW2, ALP-4.1: SW3, ALP-4.2: not available),

- the supply voltage falls below a certain threshold (all ALP-4 versions)
- the device is released (AlpbDevFree).

Note: The ALP *basic* API has no special precautions when in power-down mode. It will continue accepting commands for this device.

Voltage drops can only be reported to the API in ALP-4.2. The DMD will also enter power-down mode on other ALP-4 versions, but the APIs have no means to detect it.

One of the following actions is required in order to wake up the device from power-down mode:

- set this control bit to '0',
- use the reset switch (ALP-3: toggle SW1 to closed position and back to open position, ALP-4.1: push SW2, ALP-4.0 and ALP-4.2: not available), or
- re-allocate the device (AlpbDevFree and AlpbDevAlloc).

ALPB_DEV_DDC_SIGNALS, Bit 3: adjust reset groups
See also the document "ALP *basic* supplement" for further information about this signal

Not Available: NS_FLIP
The ALP *basic* API takes advantage of random row addressing and therefore does not require this signal. So there is no support of a control bit for the NS_FLIP signal.

Return codes

ALPB_SUCCESS
ALPB_ERR_HDEVICE
ALPB_ERR_DISCONNECT
ALPB_ERR_CONNECTION
ALPB_ERR_MT
ALPB_ERR_MEM
ALPB_ERR_PARAM
ALPB_ERR_DONGLE

6.6 AlpbDevInquire

Syntax

```
long AlpbDevInquire(ALPB_HDEVICE hDevice, long QueryType,
                    void *pValue)
```

Description

Query information about a specific device. ControlType acts as a selector whilst pValue points to a variable of the according data type.

Valid query types are a superset of the control types used in the function

AlpbDevControl

This function always checks if the device is connected to the USB, and returns ALPB_ERR_DISCONNECT if not.

Parameters

hDevice	ALP *basic* device handle (retrieve it using the function AlpbDevAlloc)
QueryType	Selection of the data to be queried
pValue	Out pointer to a memory range where output is written to

QueryType	Data type (bytes)	Description
ALPB_DEV_HALT	unsigned long (4)	see function AlpbDevControl
ALPB_DEV_DRIVER_VER	ALPB_VERSION (8)	Query version information of the device driver.
ALPB_DEV_FIRMWARE_DATE	ALPB_DATE (6)	Query version information of the USB controller firmware.
ALPB_DEV_CONFIG_DATE	ALPB_DATE (6)	Query version information of the application FPGA configuration.
ALPB_DEV_SERIAL	unsigned long (4)	Query serial number of the device.
ALPB_DEV_DMDTYPE	unsigned long (4)	Query the selected DMD type of this device.
ALPB_DEV_VERSION	unsigned long (4)	Query the hardware version. 0x0300 ALP-3 *basic* + D3000 board 0x0400 ALP-4 *basic* + D4000 board (ALP-4.0) 0x0401 ALP-4 *basic* + D4100 board (ALP-4.1) 0x0402 ALP-4 *basic* + V4100 board (ALP-4.2)

QueryType	Data type (bytes)	Description
`ALPB_DEV_DDC_VERSION`	`unsigned long` (4)	Query the DDC chipset version.
`ALPB_DEV_SWITCHES`	`unsigned long` (4)	Query the DIP switch states. Bit 0: switch number 1 is ON when '1' … Bit 7: switch number 8 is ON when '1'
`ALPB_DEV_DDC_SIGNALS`	`unsigned long` (4)	Query DDC control signals. See also 0 AlpbDevControl. Bit 0: complement image data when '1' Bit 1: disable watchdog timer when '1' Bit 2: float all DMD mirrors for power down operation when '1' Bit 3: adjust reset groups *Note*: Power down operation can also be triggered by an on-board push button. The returned value also shows this state. Use AlpbDevControl(ALPB_DEV_DDC_SIGNALS) to wake up the device again.

Table 9: Query types (AlpbDevInquire)

Return codes

ALPB_SUCCESS

ALPB_ERR_HDEVICE

ALPB_ERR_DISCONNECT

ALPB_ERR_CONNECTION

ALPB_ERR_MT

ALPB_ERR_MEM

ALPB_ERR_PARAM

ALPB_ERR_DONGLE

6.7 AlpbDevFree

Syntax

```
long AlpbDevFree(ALPB_HDEVICE hDevice)
```

Description

Release the device. Subsequent API function calls refering this device can only be done after re-allocation using the function AlpbDevAlloc.

All mirrors are released from their active position to the floating position. No further precautions have to be done prior to power off.

The multi threading error ALPB_ERR_MT can only occur when one of the functions AlpbDevLoadRows, AlpbDevClear, and AlpbDevReset is currently executed.

Parameters

hDevice ALP *basic* device handle (retrieve it using the function AlpbDevAlloc)

Return codes

ALPB_SUCCESS
ALPB_ERR_HDEVICE
ALPB_ERR_MT

6.8 AlpbDevLoadRows

Syntax

```
long AlpbDevLoadRows(ALPB_HDEVICE hDevice, unsigned char *pImage,
                     long FirstRow, long LastRow)
```

Description

Transmit rows of the bitmap *pImage to the DMD via USB. The horizontal bar (rows [FirstRow—LastRow] including) is written to the DMD without touching the other rows.

FirstRow is required to be less than or equal to LastRow. Independent of these two parameters, the pointer pImage points to the upper left pixel of the bitmap.

The data structure is compatible to the ALP *high-speed* API. It is an array of bytes. The MSB of each byte specifies the value of one pixel, 0 means black, 1 is white. The array as defined in C is:

```
unsigned char pImage[768*1024];       // XGA image
unsigned char pImage[1050*1400];      // SXGA+ image
unsigned char pImage[1080*1920];      // 1080p image
unsigned char pImage[1200*1920];      // WUXGA image
unsigned char pImage[800*1280];       // WXGA image
```

Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	Pixel	ignored	ignored	ignored	ignored	ignored	ignored	ignored

Table 10: Pixel format

For other programming languages, create your own data structure using these hints:

DMD type	**XGA**	**SXGA+**	**1080p**	**WUXGA**	**WXGA**
Size of one image	786,432 bytes	1,470,000 bytes	2,073,600 bytes	2,304,000 bytes	1,024,000 bytes
Byte offset of pixel (x, y)	y*1024 + x	y*1400 + x	y*1920 + x	Y*1920 + x	Y*1280 + x

Table 11: Image data structure

Parameters

hDevice	ALP *basic* device handle (retrieve it using the function AlpbDevAlloc)
pImage	In pointer to the bitmap data of the image to load
FirstRow	First DMD row to load
LastRow	Last DMD row to load

Return codes

ALPB_SUCCESS
ALPB_ERR_HDEVICE
ALPB_ERR_DISCONNECT
ALPB_ERR_CONNECTION
ALPB_ERR_MT
ALPB_ERR_HALT
ALPB_ERR_MEM
ALPB_ERR_MEM_I
ALPB_ERR_PARAM
ALPB_ERR_DONGLE

6.9 AlpbDevClear

Syntax

```
long AlpbDevClear(ALPB_HDEVICE hDevice, long FirstBlock,
                  long LastBlock)
```

Description

Execute the clear operation on multiple reset blocks of the DMD. Refer to the document "ALP *basic* Supplement" and the DMD data sheets for an assignment of block numbers to DMD rows.

Prerequisite: 0≤FirstBlock≤LastBlock≤15

Parameters

hDevice	ALP *basic* device handle (retrieve it using the function AlpbDevAlloc)
FirstBlock	First reset block to be cleared
LastBlock	Last reset block to be cleared

Return codes

ALPB_SUCCESS
ALPB_ERR_HDEVICE
ALPB_ERR_DISCONNECT
ALPB_ERR_CONNECTION
ALPB_ERR_MT
ALPB_ERR_HALT
ALPB_ERR_PARAM
ALPB_ERR_DONGLE

6.10 AlpbDevReset

Syntax

```
long AlpbDevReset(ALPB_HDEVICE hDevice, long ResetType,
                  long ResetAddr)
```

Description

Trigger a reset operation on the DMD. According to the DDC3000, DDC4000, and DDC4100 data sheets single block, block pair, quad block, and global reset is available. Refer to the

document “ALP *basic* Supplement” and the DMD data sheets for an assignment of block numbers to DMD rows and for valid reset groups.

Parameters

hDevice	ALP *basic* device handle (retrieve it using the function AlpbDevAlloc)
ResetType	Selection of how many reset blocks are to be reset (see table below)
ResetAddr	Selection of which blocks are reset (see table below)

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 12: Reset block assignment by (ResetType, ResetAddr)

Return codes

ALPB_SUCCESS
ALPB_ERR_HDEVICE
ALPB_ERR_DISCONNECT
ALPB_ERR_CONNECTION
ALPB_ERR_MT
ALPB_ERR_HALT
ALPB_ERR_PARAM
ALPB_ERR_DONGLE

7 Hints

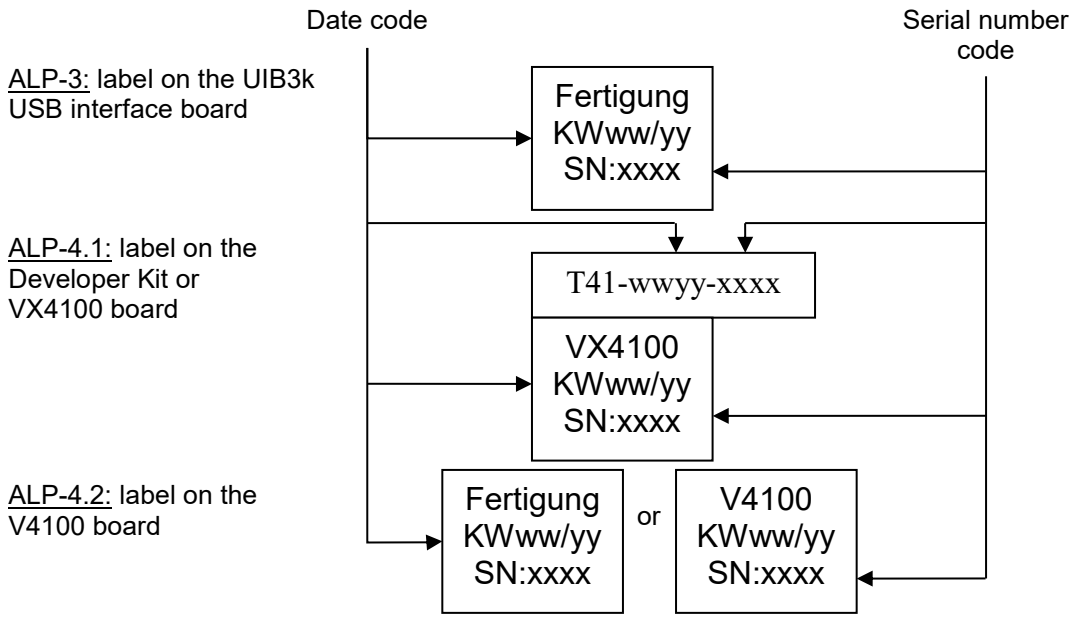
7.1 DMD type support

ALP-4 *basic* supports XGA and 1080p from the first version and WUXGA starting at ALP-4.1, version 1.0.0.5. The ALP-3 *basic* API allows to select the DMD type: XGA or SXGA+ since version 1.0.0.7. This extension changes the API behavior in the following points:

- AlpbDevAlloc: ALP-4 *basic* automatically detects the DMD type; ALP-3 *basic* sets the DMD type to ALPB_DMDTYPE_XGA by default.
  ALP-4.1 and ALP-4.2 *basic* can detect whether a DMD is connected or not. The API emulates a 1080p DMD in the case that the DMD could not be recognized or is not connected. AlpbDevInquire then returns ALPB_DMDTYPE_DISCONNECT.
- AlpbDevControl and AlpbDevInquire: additional ControlType ALPB_DEV_DMDTYPE
- AlpbDevLoadRows: image data size changes depending on DMD type
- AlpbDevClear: the set of DMD rows addressed by each block depends on DMD type
- AlpbDevReset: the set of DMD rows addressed by each block depends on DMD type
- AlpbDevReset: the set of valid reset block groups depends on DMD type

7.2 ALP Serial Number and Labeling

AlpbDevAlloc allows to select from multiple attached ALP *basic* devices by serial number. This serial number can be derived from the stickers on the hardware.



The serial number of ALP-3 *basic* is equal to xxxx.

For DLP® Discovery™ 4100 Developer Kits please use ALP-4.1 *basic* and add 5000 to the serial number code - so use DeviceNum=5000+xxxx for device allocation.

The serial number of ViALUX VX4100 (ALP-4.1 *basic*, V-9500/V-9600 Modules) as well as ViALUX V4100 (ALP-4.2 *basic*, V-7000 Modules) is equal to xxxx.

7.3 DMD Float operation

All mirrors are released from their active position to the floating position automatically when an ALP *basic* device is released (AlpbDevFree). This eliminates the need to push a button prior to power off.

Note: Only the DLP® Discovery™ 3000 board requires to push the button SW2 before shutdown, when used without ALP.