VALCam PTZ SDK Documentation

Table of Contents

Introduction	2
Using the SDK	4
Basic Application Structure	5
Init System	16
Send Window Handle	18
Set Cropping Parameters	19
Display Video	20
Scale Captured Image	21
Freeze	22
No Video Display	23
Auto White	24
NEW Accurate Auto White	25
Live Capture	26
Flash Capture	28
Display Capture	30
Color Correct Image	31
Save Captured Image	32
Discard Captured Image	33
Message Drain Handle	34
Set Flash Intensity	35
Close Card	36
Rotate Image	37
Set Zoom Position	38
NEW Set Zoom Array	39
Read Zoom	40
Standard Zoom Controls	40
Center Camera	41
Set Pan/Tilt Position	42
Zoom Controls	43
Live Exposure Control	44
NEW Return Live Exposure	46
Live Color Settings	47
Flash Color Settings	48
Set Live Exposure	49
Set Flash Color Correction	50
Read Status Info	51
New Read All Staus Info	52
Set Live Defaults	53
Set Flash Defaults	54
Set Frame Grabber Sharpness	55
NEW Buffer Capture Function	54
Flash Distance Settings	55
NEW Capture Structure Function	57
NEW Capture Structure Buffer Function	58
Flash Iris Distance Settings	59
Registery Info	60
Hints and Tips	61
SDK Definitions	62
Exported Function Definitions	65

Documentation for the VALCam USB Software Developers Kit

Introduction

The Software Developers Kit for the **VALCam** image acquisition system is designed for quick, easy integration into applications developed for **Win 2000/XP/VISTA/Win7**. The primary design goal has been a simple, easy to use SDK without any reduction of power or scope. All low level details are handled invisibly in the DLL to reduce complexity at the integrator level while maintaining full power and versatility.

Any development platform that allows DLL function calls can use this SDK. Simple, full featured code examples are provided in Visual Basic, Visual C++, .NET, and Delphi. The simplicity of the SDK illustrated in the code examples will assist the developer in integrating the camera quickly and efficiently.

SDK and Hardware Features

- Live preview display, image capture, synchronized strobe flash, and camera zoom control are all integrated through one DLL function call.
- Support for immediate camera positioning. Automatic face centering capabilities. (PTZ model only)
- Restoring and saving of all state information is done automatically.
- The synchronized flash provides a consistent light source for high quality natural photo-images in any environment, even dark rooms.
- Software based image control to fine-tune the system for optimal results in any configuration.
- Software control of camera position (pan, tilt, zoom) for increased operator efficiency and through put.
- Software calls optimized for both live display and flash image capture.
- SDK support for preview of captured images
- Color correct images in real time
- Can be integrated with any development platform that supports DLL function calls.
- Mouse control of pan, tilt, (PTZ unit only) zoom and exposure.
- Uses optical zoom for highest resolution.
- Software control of flash intensity for optimal image exposure under any lighting conditions

- Preview and Capture in portrait mode to match photo-id aspect ratios.

Hardware Installation

Follow the hardware and software installation instructions in the **VALCam User Manual**.

Verify that the system is operating properly according to the description in the user manual before proceeding.

SDK Software Installation:

Run the **VALCam SDK Install.exe** program. This installation script will install the **USB_SDK.Dll** and all necessary files. This install contains all files needed to for standalone application developement

Sample SDK Tester programs will be installed stored in the **VALCam PTZ SDK** group under **Start->Programs->VALCam PTZ SDK .**

Two examples written in C# and C++ (Win32) are provided. Source code for these examples is provided.

//*
NOTE: Always disconnect power to the camera before installing updated SDK or camera software. New Camera firmware will not be updated unless the camera is powered cycled after installing new software. This only applies to software updates from Video Associates Labs. Development code can be installed without powering down the camera.
//***

DirectX Requirements

The SDK requires a minimum of DirectX 8.1. All latter versions of DirectX are compatible. The most recent DirectX updates are available at no charge at www.microsoft.com.

The following C++ code can be used to determine the Direct Version number:

```
DWORD version, revision;
DirectXSetupGetVersion(&version,&revision);
  version=version & 0x0000000F;
  if (version<8){
    MessageBox("TEST");
    MessageBox(TEXT("DirectX version must be  8.1 or greater.\r\n")
      TEXT("Install the DirectX upgrade provided on the installation CD.\r\n")
      TEXT("Restart the Camera Application after Direct X has been upgraded.\r\n"),
      TEXT(""), MB_OK | MB_ICONINFORMATION);
    exit(2);
  }
```

The Microsoft DirectX system file**, DSETUP.dll**, must be located in the search path for this function to be used. This file is provided in the SDK install script

The Direct X version number can also be obtained from **Start->Programs->Accessories->System Tools ->System Information.**

.The latest Microsoft DirectX runtime is available for download on their web site and can be freely distributed.

Using the SDK

All device and camera control is accessed through 4 exported functions located in **USB_SDK.Dll**

The Function call is:

int VALCam_Control(int Message, int Param1, int Param2, int Param3, int Param4) //C++

The first parameter, **Message**, is a command that instructs the system to perform an action. Typical commands might be zoom, pan right, display video, capture, capture with flash and so on. The other parameters provide information on how this command is to be carried out. For example how fast to zoom, where to display video, what format to store the capture and so on.

All constants and definitions are defined in **USB_SDK Definitions.h** supplied with the SDK.

This function will need to be exported into your development platform.

Examples:

C# --

```
[DllImport("USB_SDK.Dll")]
public static extern int VALCam_Control(int Msg,int
Param1,int Param2,int Param3, int Param4);
```

C++--

```
extern "C" _declspec(dllexport)int CALLBACK
VALCam_Control(int, Message,int Param1,int Param2, int
Param3, int Param4);
```

Visual Basic 6

```
Private Declare Function VALCam_Control Lib "USB_SDK.DLL"
(ByVal Message As Long, ByVal Param1 As Long, ByVal Param2
As Long, ByVal Param3 As Long, ByVal Param4 As Long) As
Long
```

Delphi

```
Function VALCam_Control
(Msg,Param1,Param2,Param3,Param4:Integer):Integer;stdcall;
external  'USB_SDK.Dll' name 'VALCam_Control';
```

Additional New exported function calls include:

```
extern "C" _declspec(dllexport)int CALLBACK
VALCam_Capture(prtstructCapture CaptureStruct)

extern "C" _declspec(dllexport)int CALLBACK
VALCam_ReadReturnParameters(ptrstructReturnValues
ReadReturnParameters)

extern "C" _declspec(dllexport) int CALLBACK
VALCam_CaptureToBuffer(unsigned char * ImageBuffer, int Preview,
int Flash )

extern "C" _declspec(dllexport) int CALLBACK
VALCam_CaptureToPreviewBuffer(unsigned char * ImageBuffer)
```

These calls are discussed latter in the documentation.

Basic structure of an application using the SDK.

For details on all commands please refer to the individual command descriptions.

Step 1 – Initializing the DLL and camera hardware.

The first call to the DLL **must** be the **VC_InitSystem** command.
This command performs the following functions:

- Initializes software system structures
- Initializes camera hardware
- Reads the registery entries file and initializes variables from defaults or previous values
- Returns a value<1 if an error is detected.
- Returns a value >0 indicating the camera hardware detected.
- Optionally passes a Window handle, in **Param1** that will be used to display the live video stream after the **VC_Display** command.

This command should only be issued once during the life of an application as it takes a couple of seconds to complete.
Other commands can be used to release system resources without having to close and reissue another **VC_InitSystem** command.

Example:

int errorlevel=**VALCam_Control(VC_InitSystem,0,0,0,0,)** or
int errorlevel=**VALCam_Control(VC_InitSystem,m_hWnd,0,0,0)**.

Notes:

This command should only be issued once per application as it can take a few seconds to complete.

When the camera is not in use, the **VC_Freeze** command can be used to remove the live video display and free system resources.
To put the camera in use, use the **VC_Freeze** command to display live video and enable camera functionality.

Before closing the application the **VC_CloseCard** command must be issued. If this command is not issued resources will not be freed and the system hesitate the next time the camera is initialted

Step 2 - Displaying the live video stream.

The following sequence of commands should be used to display live video:.

1. If a window handle was not passed as **Param1** in the **VC_InitSystem** command, the **VC_SetWindowHandle** command must be used to pass the window handle used to display the video stream.

   VALCam_Control(VC_SetWindowHandle, (int) m_hWnd,0,0,0)

   m_hWnd will be the window used to show live video.
   This command only needs to be called when changing the window handle used for live video.

2. Use the **VC_SetCropParameters** command to set starting and ending horizontal and vertical pixel positions for the live video stream and captured image.

   If this command is not issued, default values are used.

   This command can be used to crop the video stream to match the aspect ration of the picture box on the card design. This allows previewing and capturing the image in the same aspect ratio used for printing. This command can also be used to eliminate the black borders that can surround the video when displaying the full 480 X 640 video stream.

   VALCam_Control(VC_SetCropParameters, 3, 5, 477, 635)

   Param1 (3) indicates that the 3rd pixel on each line is the 1st pixel displayed. The first 2 pixels are discarded.
   Param3 (477) indicates that the next 477 pixels are displayed.
   NOTE: Param1 + Param3<=480, or the command is ignored

   Param2 (5) indicates that the 5th line is the first line displayed. Lines 1 - 4 are discarded.
   Param4 (762) indicates that the next 762 lines are displayed.
   NOTE: Pamam2 + Param4 <=640 or the command is ignored.

3. Use the **VC_DisplayVideo** command to show the live video stream.

   VALCam_Control(VC_DisplayVideo, 20,40, 477,635)

   This command will start the live video stream at row 40 and column 20 on the Window passed in the **VC_SetWindowHandle** or **VC_InitSystem** command. The displayed video will be scaled to 477 X 635.

   VALCam_Control(VC_DisplayVideo,0,0,286, 381)

   This command will start the live video stream at row 0 and column 0 on the Window passed in **VC_SetWindowHandle** or **VC_InitSystem** command. The displayed video will be scaled to 286 X 381.

   NOTE: **Only the live video will be scaled, the captured video size is determined by Param3 and Param4 of the VC_SetCropParameters command.**

4. Use **VC_FreezeVideo** to freeze, unfreeze and make the live video stream invisible.

NOTE: This command can also be used to free system resources when not using the camera.

This instruction will stop the live video stream and freeze the last video frame on the screen. Since the video stream is stopped, system resource are freed but the DLL is still initialize.

VALCam_Control(VC_Freeze,0,0,0,0)

This instruction will stop the live video stream and clear the screen**.** Systems resources are freed, but the Dll is still initialized.

VALCam_Control(VC_Freeze,0,1,0,0)

This instruction will unfreeze and start the live video stream.

VALCam_Control(VC_Freeze,1,0,0,0)

Step 3 - Capturing a still image

1. Verify that the camera system has been initialized and assigned to a video window. The video should be unfrozen, and must be assigned to a window handle. See **VC_SetWindowHandle** and **VC_InitSystem** command.

2. This command will capture a live video frame:

   VALCam_Control(VC_LiveCapture, Sharpness,Preview,0,CaptureType)

   The **"Sharpness"** value will determine the amount of post capture sharpness added to the image. This value can vary from 200 – 999.

   If **Preview=0** the image will be captured and saved immediately

   If **Preview=1** the image will be temporarily saved in a format for viewing. The **VC_DisplayCapture** command can then be used to display the capture in a specific window at a specific location.

   The **CaptureType** value will determine how the image is stored.
   Options are:

   a. DIB =0
   b. DDB (for .NET) =1
   c. Clipboard =2
   d. BMP =3
   e. JPG >=4

   NOTE:**If the image is saved as DIB or DDB, the application must free the image after processing or a memory leak will occur. See SDK Tester souce code for examples**

   Use **VC_SetLiveRed, VC_SetLiveBlue,VC_SetLiveColor, VC_SetExposure, VC_LIveIrisUp, VC_LiveIrisDown, VC_Brightness and VC_Contrast** to adjust exposure and color settings **BEFORE** taking captures.

   The **VC_ColorCorrectImage** command can then be used to color correct the image in real time with scroll bar controls in the application. This process is very efficient for adjusting image color balance in real time after a capture while viewing the changes in real time.

   The **VC_SaveCaptureImage** can then be used to save the image as specified in the **CaptureType** parameter of the **VC_LiveCapture** command.

   See the SDK application demo for the use of these commands.

Capture an image using the Flash

This command will use the flash to capture an image.

VALCam_Control(VC_FlashCapture, Sharpness,Preview, 0 ,CaptureType)

This command will initiate a flash capture.

The intensity of the flash is determined by the **FlashIrisLevel** internal parameter. This parameter is set by the **VC_SetFlashIrisLevel** command. This value depends on the subject distance from the camera and must be optimized during initial setup.

Refer to the doc. for suggested values at different subject distances. The greater the value the brighter the flash intensity.

If **Preview=0** the image will be captured and saved immediately

If **Preview=1,** the image will be temporarily saved in a format for viewing by using the **VC_DisplayCapture** command. The image can then be displayed in a specified window at a specific location. The **VC_ColorCorrectImage** command can then be used to color correct the image in real time with scroll bar controls in the application. This process is very efficient for adjusting image color balance in real time after a capture.

The **VC_SaveCaptureImage** can then be used to save the image as specified in **Param 4** of the **VC_LiveCapture** command.

See the SDK application demo for the use of these commands.

Use **VC_SetFlashRed, VC_SetFlashBlue,VC_SetFlashColor, VC_SetFlashIrisLevel, VC_Brightness and VC_Contrast** to adjust exposure and color settings **BEFORE** taking a capture.

Exposure adjustments should be controlled primarily by **VC_SetFlashIrisLevel.** Small changes can be made to **VC_SetBrightness** and **VC_SetContrast** to fine tune exposure.

NOTE:
A new exported function call can be used to initiate captures:
VALCam_Capture(ptrCaptureStructure)

This command replicates the capture features discussed earlier.
Details on this call or discussed latter in the doc.

The supplied source code has examples using this call.

NOTE:
A new call, **VALCam_Capture(ptrCapStruc pCaptureStructure)**, has been added that can be used to replace this call.

Step 4 Control Camera Zoom Function

1. Use **VC_ZoomIn, VC_ZoomOut, VC_ZoomInFast, VCZoomOutFast, VC_ZoomInSlow, VC_ZoomInFast and VC_ZoomArray.**

   These commands will zoom the camera in or out at the selected speed. The camera will continue to zoom until a **VC_ZoomStop** command is issued.

   Example:

   VALCam_Control(VC_ZoomInFast,0,0,0,0); //Camera will zoom in fast
   Sleep(1000);
   VALCam_Control(VC_ZoomStop,0,0,0,0); //Camera will stop zooming.

` 2. Use **VC_SetZoomPosition** to immediately place the camera at a specific zoom position.

   Refer to the command description for a list of values and zoom magnifications.

   Example :

   VALCam_Control(VC_SetZoomPosition,0,0,0,0); // Sets camera to zoom wide
   VALCam_Control(VC_SetZoomPosition,0X1606,0,0,0); // Sets camera to zoom at x2.

3. Use **VC_ReadZoomPosition** to read the current zoom position.

   Example:

   int CurrentZoomPosition;
   CurrentZoomPosition = VALCam_Control(VC_SetZoomPosition,0,0,0,0)

4. Use **VC_ZoomArray** to immediately set zoom from 1 -50 setting at equal 2% magnificaton increments.

   Example:

   int ZoomArrayValue=25;
   VALCam_Control(VC_ZoomArray, ZoomArrayValue,0,0);

Step 5 Control Pan/Tilt Position (Only available on PTZ model)

1. Use **VC_PanLeft** (Fast, Slow)**, VC_PanRight** (Fast, Slow)**, VC_TiltUp** (Fast, Slow)**, VC_TiltDown** (Fast, Slow)**, VC_PanTiltStop.**

   These commands will pan and tilt the camera at the selected speed. The camera will continue to pan/tilt until a **VC_PanTiltStop** command is issued.

   Example:

   VALCam_Control(VC_TiltUpFast,0,0,0,0); //Camera will tilt up fast
   Sleep(1000);
   VALCam_Control(VC_PanTiltStop,0,0,0,0); //Camera will stop moving.

2. Use **VC_SetPTZPosition** to immediately position the camera to any position immediately.

   This powerful command can be used to implement automatic face finding.

   Example:

   VALCam_Control(VC_SetPTZPosition,34,-8,0,0)

   This command will move the camera 34 pixels to the right (as viewed in the monitor) and 8 pixels down (as viewed in the monitor)

Step 6 Responding to Window Messages on the live preview display

The **VC_MessageDrainHnd** command allows setting an application window to respond to window messages on the live preview window. This capabilitiy aids in allowing the application to respond to mouse clicks.

See the sample application for an example of using this command to create “Double Click” (PTZ version) autocentering and click, drag and release for immediate camera positioning.

Example:

```
CMouseCapture1.Create(NULL,"MC",0,rect,this,0);
CMouseCapture1.Ptr=this;
VALCam_Control(VC_MessageDrainHnd,(int)CMouseCapture1.m_hWnd,0,0,0);
```

The CMouseCapture1 class is now associated with the “Live Video” window and will respond to mouse clicks.

Step 7 Immediate Preview and ColorCorrection of the captured image

If the **VC_LiveCapture** or **VC_FlashCapture** commands are issued with **Param2=1**, a temporary image is created in memory. This image can be displayed in an application window for preview and modification.

Example:

Preview=1;
VALCam_Control(VC_FlashCapture,FlashIrisLevel,Preview,0,1)
VALCam_Control(VC_DisplayCapturedImage,(int) hWnd, 0,0)

These commands will capture the image to memory and display the image on hWnd beginning at column 0, row 0.

VALCam_Control(VC_ColorCorrectImage,20,-5,0,0);
VALCam_Control(VC_DisplayCapturedImage,(int) hWnd, 0,0)

This command will increase the red value of the image by 20 and decrease the blue value by 5. The color corrected image will then be redrawn on the display with the new color changes.

VALCam_Control(VC_SaveCapturedImage,0,0,0,0)

This command will save the image according to the value **Param4** in the **VC_FlashCapture** command. In this example the image will be saved as a DIB.

The image can also be discarded by using the **VC_DiscardCapturedImage.**

In any case, **VC_SaveCapturedImage** or **VC_DiscardCapturedImage** must be called before another capture command can be issued.

Note that the image can be captured and saved immediately by using the **VC_LiveCapture** or **VC_FlashCapture** commands with **Param2** =0.

Step 8 Disconnecting live video stream from window handle

If the application destroys the window used for the live video display the **VC_NoVideoDisplay** command should be used to disconnect the live video stream from the window before the window is destroyed

`

Using this command will insure that proper handling of software structures are maintained. **Do not destroy the window used to display video without first calling this command.**

If you need to make the live video invisible, but do not need to destroy the window use the **VC_Freeze** command as discussed in **Step 2 section 4.**

Example:
VALCam_Control(VC_NoVideoDisplay,0,0,0,0);
The live video stream is no longer associated with a window.

NOTE: Be sure to reconnect the live video stream to a window handle before trying to display the live video stream.

Step 9 Unloading the DLLand releasing all resources

Use the **VC_CloseCard** to unload the SDK drivers and release all resources.

The camera cannot be used again until a **VC_InitSystem** is issued.

This command should be called before exiting your application or when the SDK functions will not be needed for the duration of the application.

NOTE:
Failure to call this command when exiting your application can cause a failure.

During development if you must exit your code before executing this call, unplug the USB cable to the PC before you issue the VC_InitSystem call again.

NOTE:
It recommended to only use this command when exiting your application. Otherwise additional VC_InitSystem will have to be called each time when using the SDK. Since this command can take to few seconds to complete its more efficient to use the VC_Freeze command when not using the camera system.

Storing and Retrieving State Parameters

All state parameters are written and read from the registery when the DLL is closed and initiated. This value are written to: **HKEY_CURRENT USER\SOFTWARE\ VALCam USB SDK Zoom**

These values can be read individually through various VC_ReturnXXX commands.

A new call **VALCam_ReadReturnParameters(ptrStruc ReadRetrunStruc)** will return all value in a structure passed from the application.

Refer to the doc and sample source codes files for usage.

SDK Command Descriptions

Initializing the Camera

VC_InitSystem = 0
The first command sent to the DLL must be **VC_InitSystem. The system will not respond until this command is sent.** This command also optionally passes the handle of the window to display live video.

The **Registery** is also read and internal variables are updated.
Registery values are store and written to:
HKEY_CURRENT USER\SOFTWARE\ VALCam USB SDK Zoom

A value <=0 is returned if this function fails. A positive value indicates success

Example:
VALCam_Control(VC_InitSystem,0,0,0,0) // Window handle not sent. Must use VC_SendWindowHandle

VALCam_Control(VC_InitSystem, (int) m_hWnd,0,0,0) // Window handle sent.

This initializes the camera system. It has **to be the first call made or other messages will not work. Param1** is the handle of the window to display live video.

Params
Param1 must be set to the handle of the window that is to display the live video or 0. If **Param1** is 0 the window handle must be passed in **VC_SetWindowHandle. Note that live video will not appear** until the **VC_DisplayVideo** command is sent.

Params 2 - 4 =0.

Details
In addition to initializing the DLL, this call will read the **REGISTERY**file and restore all saved parameters. This relives the application of restoring this data. Refer to other command descriptions for the INI file entries created. Also refer to the discussion of the registery entries at the end of this document.

Constant
0

Returns
<=0 indicates error.
>0 indicates sucess

Registery Entires: None

Example
CameraType=VALCam_Control(VC_InitSystem, (int) winHnd,0,0,0)

NOTE:

This command should be called only once during the life of your application. This command should not be called again until after an VC_CloseCard command has been issued.

During development you might need to issue this command before executing the VC_CloseCard command. In this instance its recommended to unplug and reattach the USB cable.

Set Window Handle to display live video

VC_SendWindowHandle = 6

This command sets the handle of the window to display live video. The new window handle is passed in **Param1.** This command does not need to be used if the **Window handle** was passed in **Param1** of the **VC_InitSystem** command.

A **VC_DisplayVideo** command must be sent after this command or the video will not be displayed in the new window.

NOTE:
This command is only needed when a new window is used to display the live video stream.

Param 1

Handle of new window to display live video

Param 2-4

0

Registery entries: None

Return

<1 indicates error.

Example:

VALCam_Control (VC_SendWindowHandle, (int) m_hWnd,0,0,0)

Registery entries:

None

Notes

See use in example code.
This command allows changing the window that displays live video without calling **VC_InitSystem**

Set Cropping Rectangle

VC_SetCropParameteres = 156

This message sets the crop rectangle of the displayed and captured video.
The live video display will be scaled in relation to these parameters.
See **VC_DisplayVideo**

Params

Param1 – X origin of the crop rectangle
Param2 – Y origin of the crop rectangle
Param3 - Pixel width of the crop rectangle
Param4 – Line height of the crop rectangle

Returns

<1 indicates error condition.

Example:

VALCam_Control(VC_SetCropParameters,3,5,573,762)

The first pixel displayed (and captured) will be the 3rd pixel. Other pixels will be discarded.
The total pixels displayed (and captured) will be 572.
The first line displayed (and captured) will be the 5th line. Other lines will be discarded
The total lines displayed and captured will be 762.

NOTE: Parma1 + Param3 <=480
Param2 + Param4 <=640
If these conditions are not true, this command will not work.

Registery entries:

CropStartX= Param1
CropStartY=Param2
CropWidth=Param3
CropHeight=Param4

Notes

If this call is not used, default values are used to display complete image
Maximum size of the uncropped rectangle is 480 X 640. This call controls the amount of pixels provided from the camera to the computer. **VC_DisplayVideo** uses these pixels to display the video stream on the monitor. **VC_ScaleCapturedImage** uses these pixels to scale the captured image.

Display Live Video on Screen

VC_DisplayVideo=1

This message displays the live video window at a given size and position to the window whose handle was passed as **Param1** in the **VC_InitSystem** or the **VC_SendWindowHandle** command

Params

Param1 - X position (upper left corner) of the video window.
Param2 - Y position (upper left corner) of the video window
Param3 - Width of the live displayed (not captured) video window. Max value of 480
Param4 - Height of the live displayed (not captured) video window. Max value of 640

Constant

1

Registery entries:

None

Example:

VALCam_Control(VC_DisplayLiveVideo,10,20,320,240)

This command will display the live video starting at 10th column and 20th row of the window used for display. This window was set earlier using the **VC_SendWindow** command or the **VC_InitSystem** command

The size of the displayed video is scaled to 320/240.

Note: if **Param3/Param4** of **VC_SetCropParameters** does not = **320/240** the image will be distorted.

Returns

<1 indicates error condition.

Notes

VC_DisplayVideo only effects live video displayed on the monitor. This command has no effect on the captured image

The video will be scaled in relation to the Crop Parameters. See **VC_CropParameters**

Caution:

If **Parma3/Param4** used in **VC_CropParameters** does not = **Param3/Param4** of this command, the displayed video will be distorted.

I

Set Scaling Parameters

VC_ScaleCapturedImage=18

This message sets the scaling parameters of the captured image. **This call does not effect the scaling parameters of the displayed image**

The captured image will be scaled to these parameters

If either the width or height parameter is 0 scaling will not be applied. The native raw pixel format as specified in **VC_SetCroppingRectangle** will be used.

Params

Param1 – Scaled width parameter // if =0, no scaling will occur
Param2 – Scaled height parameter //if = 0, no scaling will occur.
Param3 - NA
Param4 – NA

Example:

VALCam_Control(VC_ScaleCapturedImage, 320,240,0,0)

This command will scale the captured image (not the live display) to 320 X 240.

NOTE: If **Param3/Param4** of **VC_SetCropParameters** does not = 320/240, the captured image will be distorted.

REGISTERYentries:

ScaledImageWidth
ScaledImageHeight

Returns

<1 indicates error condition.

Notes

If Param1 or Param2 = zero no scaling will occur.
If Param3/Param4 of VC_SetCropParameters is not equal to Param1/Param2 of this call the scaled video will be distorted.

Freeze Video, UnFreeze Video, Stop Video Display

VC_Freeze=2

This command has 3 modes of operation.

1. Freeze the live display and free system resources.
2. Stop displaying live video and free system resources.
3. Make live video visible

Params

Param 1 = 0 and Param2 = 0 will freeze the last frame of live video stream to screen.
Param 1 = 0 and Param2 = 1 will make video the invisible and free system resources
Param 1 = 1 will start live video stream to screen

Example

VALCam_Control(VC_Freeze,0,0,0,0) //Freeze live video and free system resources
VALCam_Control(VC_Freeze,0,1,0,0) //Live video will become invisible, and free system resources freed
VALCam_Control((VC_Freeze1,0,0,0) //Live video will be unfrozen**.**

REGISTERYentries:

None

Returns

<1 indicates error condition

Notes

In addition to stopping and starting live video, this command will free system resources when stopping the video stream (Param1 = 0). Use this command to free resources when the camera is not needed.

Stop Video Display/ Disassociate Video Stream from a Window

VC_NoVideoDisplay=3

This command should only be used when disassociating the video stream from a window. **This should be called before destroying a window that was used to display the live video stream or an error condition might be created**.

The **VC_SendWindowHandle** command should be called after this command to display the video in another window.

This command will disconnect the window used for display with live video stream. If a window is not connected with the video stream before using a **VC_DisplayVideo** or **VC_Capture** command, unpredictable results can occur.

Example:

VALCam_Control(VC_NoVideoDisplay,0,0,0,0)

Continual Auto White Balance

VC_AutoWhite=118

This command issues continual White Balance of the Live video. Only the live display and live captures are affected. This command does not affect flash captures.

Params

Param 1-4 =0

REGISTERYentries:

AutoWhite=1 //AutoWhite is On
AutoWhite=0// AutoWhite is Off
RedAutoWhite=200
BlueAutoWhite=197

Example:

VALCam_Control(VC_AutoWhite,0,0,0,0) // Continual AutoWhite is enabled

VALCam_Control(VC_AutoWhite,1,0,0,0) // Continual AutoWhite is disabled

Returns

<1 indicates error condition

Notes

When White Balance is on, the camera is continually adjusting color balance depending on the image content. Overall balance can change depending on screen content. Although this works well in most situations it does not provide the most accurate results in all situations.

It recommended to turn AutoWhite off, after proper white balance. When this is done color settings are locked and will not change even when the camera is powered cycled.

Adjusting VC**_SetRedLive** and **VC_SetBlueLive** turns AutoWhite off and locks the color to these new values. The new values are saved to the ini file and will be restored will be restored automatically at the next **VC_InitSystem**

Note

A new command **VC_AccurateAutoWhite** provides more accurate results and is recommended.

NEW - Accurate Auto White Balance

VC_AccurateAutoWhite=125

This command will perform a very accurate white balance by placing a white or gray object in front of the camera before issuig the command. The camera should view the white/gray object until the command returns.

The command will store new color value in the registery. After this command is properly issued the accurate white balance will be maintained even when the camera is power cycled.

If room lighting does not change this command will not need to be repeated.

Params

Param 1-4 =0

Registery Entries:

Auto White = 0

Notes

To accurately White Balance the camera use this procedure:

- Place a sheet of white paper in the positon of the subject.
- Zoom in camera to white paper. Adjust VC_Brightenss for proper illumination.
- Initialte the **VC_AccurateAutoWhite** command.
- When the command returns remove the white/gray object

Capture Image without flash

VC_LiveCapture=4

This command will take an immediate live capture. The image can be saved as a DIB, JPG, BMP or stored on the Clipboard. **Param1** sets the captured image sharpness.

If **Param 2=1**, the image is not saved but stored in memory for preview and color correction. If this option is used, **VC_SaveCapturedImage** or **VC_DiscardCapturedImage** must be issued before the system will respond to another **VC_LiveCapture** or **VC_FlashCapture** command

NOTE:

A new call, **VALCam_Capture(ptrStruct pCaptureStruc**) performs these same funtions

Params

Param 1 = Sets the Sharpness of the capture image. The value can range from 0 to 999. A value of 0 will default to a setting of 600.

Param 2 =1 Image is saved in a format for immediate display. The image is not saved until the **VC_SaveCapturedImage** is called. The option is use full for immediately previewing the capture. Color correction commands can be used in real time to modify the image before saving. See **VC_DisplayCapture**, **VC_ColorCorrectImage**, **VC_SaveCapturedImage**, and **VC_DiscardCapturedImage**

Param 2=0 The image is saved immediately according to **Param 4**

Param 4 =0 Returns handle to DIB in memory. **Note:The application must free this memory**

Param 4 =1 Returns handle to DDB in memory used for **.NET** apps
Note: The application must free this memory.

Param 4 = 2 Puts image on Clipboard

Param 4 = 3 Saves Image as “Image4.BMP” in the current directory

Param 4 >=4 Saves Image as “Image4.JPG” in the current directory with a quantization factor of the value of **Param 4**
Higher values of Param4 result in smaller file sizes and decreased image quality. Generally values <25 provide for outstanding image quality.

REGISTERYentries:

Sharpness (Param1)

Notes:

Param1 controls image sharpness. Higher values result in a sharper image but if the value is too high the image will become edgy and grainy. The user is encouraged to experiment with different values.
When saving to a DIB (Param4=0) or DDB (Param4=1) a return value of 0 indicates an error. All other values are valid.

Returns

< 1 indicates error condition for Clipboard, JPG, and Bmp captures. (Param 4= 3,4,5) =0 indicates an error of DIB and DDB captures. All other values are valid.

ERROR_SEQUENCE	-1	Sequence error
ERROR_NO_BITMAP	-2	Invalid bitmap handle
ERROR_MEMORY_TOO_LOW	-3	Not enough memory available
ERROR_FILE_LSEEK	-4	Error seeking to position
ERROR_FILE_WRITE	-5	Error writing file
ERROR_FILE_GONE	-6	File not present - abort
ERROR_FILE_READ	-7	Error reading file
ERROR_INV_FILENAME	-8	Invalid filename specified
ERROR_FILE_FORMAT	-9	Invalid file format
ERROR_FILENOTFOUND	-10	File not found
ERROR_INV_RANGE	-11	Invalid width/height
ERROR_IMAGE_TYPE	-12	Image format recognized, but sub-type not supported
ERROR_INV_PARAMETER	-13	Invalid parameter passed
ERROR_FILE_OPEN	-14	Not able to open file
ERROR_UNKNOWN_COMP	-15	Unknown compression format
ERROR_FEATURE_NOT_SUPPORTED	-16	Feature not supported
ERROR_NOT_256_COLOR	-17	VGA card only supports 256 colors (8 bit)
ERROR_PRINTER	-18	Printer error
ERROR_CRC_CHECK	-19	Data CRC check error
ERROR_QFACTOR	-21	Invalid QFactor specified
ERROR_TARGAINSTALL	-22	TARGA not installed
ERROR_OUTPUTTYPE	-23	Invalid compression format
ERROR_IMAGE_Exists	-100	Previous image not deleted

Capture Image with Flash

VC_FlashCapture = 7

This command will take a flash capture. If **Param2 =0** the image can be saved as a DIB, JPG, BMP or stored on the Clipboard. The Flash intensity is set by the **VC_SetFlashIrisLevel** command

If **Param 2=1**, the image is not saved but stored in memory for preview and color correction. If this option is used, **VC_SaveCapturedImage** or **VC_DiscardCapturedImage** must be called to release the image before another capture command is allowed.

NOTE:

A new call, **VALCam_Capture(ptrStruct pCaptureStruc**) performs these same funtions

Params

Param 1 = Sets the Sarpness of the capture image. The value can range from 0 to 999 A value of 0 will default to a setting of 600.

Param 2 =1 Image is saved in a format for immediate display. The image is not saved until the **VC_SaveCapturedImage** is called. The option is use full for immediately previewing the capture. Color correction commands can be used in real time to modify the image before saving. See **VC_DisplayCapture**, **VC_ColorCorrectImage**, **VC_SaveCapturedImage**, and **VC_DiscardCapturedImage**

Param 2=0 The image is saved immediately according to **Param 4**

Param 4 =0 Returns handle to DIB in memory. Note:The application must free this memory

Param 4 =1 Returns handle to DDB in memory used for DOT NET apps
Note: The application must free this memory.

Param 4 = 2 Puts image on Clipboard

Param 4 = 3 Saves Image as “Image4.BMP” in the current directory

Param 4 >4 Saves Image as “Image4.JPG” in the current directory with a quantization factor of the value of Param 4
Higher values of Param4 result in smaller file sizes and decreased image quality. Generally values <25 provide for outstanding image quality.

REGISTERYentries:

Sharpness (Param1)

Notes:

The Flash intensity is set by **VC_SetFlashIrisLevel**. Flash intensity is set according to subject distance from the camera. Once this value is set it will not need to be changed **Param1** sets image sharpness. Higher values result in a sharper image but if the value is too high the image will become edgy and grainy

If **Param2 =1**, the image will be available in memory to be displayed (**VC_DisplayCapture**), color corrected (**VC_ColorCorrected**), saved (**VC_SaveCapturedImage**) or discarded (**VC_DiscardPreviewImage**).

Returns

<1 indicates an error condition for Clipboard, JPG and BMP
=0 indicates an error for DIB and DDB captures. All other values are valid.

.

See **VC_LiveCapture** for error codes. See **VC_SaveCapturedImage** and **VC_DiscardCapturedImage**

ERROR_SEQUENCE	**-1**	**Sequence error**
ERROR_NO_BITMAP	**-2**	**Invalid bitmap handle**
ERROR_MEMORY_TOO_LOW	**-3**	**Not enough memory available**
ERROR_FILE_LSEEK	**-4**	**Error seeking to position**
ERROR_FILE_WRITE	**-5**	**Error writing file**
ERROR_FILE_GONE	**-6**	**File not present - abort**
ERROR_FILE_READ	**-7**	**Error reading file**
ERROR_INV_FILENAME	**-8**	**Invalid filename specified**
ERROR_FILE_FORMAT	**-9**	**Invalid file format**
ERROR_FILENOTFOUND	**-10**	**File not found**
ERROR_INV_RANGE	**-11**	**Invalid width/height**
ERROR_IMAGE_TYPE	**-12**	**Image format recognized, but sub-type not supported**
ERROR_INV_PARAMETER	**-13**	**Invalid parameter passed**
ERROR_FILE_OPEN	**-14**	**Not able to open file**
ERROR_UNKNOWN_COMP	**-15**	**Unknown compression format**
ERROR_FEATURE_NOT_SUPPORTED	**-16**	**Feature not supported**
ERROR_NOT_256_COLOR	**-17**	**VGA card only supports 256 colors (8 bit)**
ERROR_PRINTER	**-18**	**Printer error**
ERROR_CRC_CHECK	**-19**	**Data CRC check error**
ERROR_QFACTOR	**-21**	**Invalid QFactor specified**
ERROR_TARGAINSTALL	**-22**	**TARGA not installed**
ERROR_OUTPUTTYPE	**-23**	**Invalid compression format**
ERROR_IMAGE_Exists	**-100**	**Previous image not deleted**

Displaying, Saving, Color Correcting and Discarding Captured Images

VC_DisplayCapture = 22

If **VC_LiveCapture** or **VC_FlashCapture** was used **with Param2 (Preview**) = 1, then the image was saved in a format the can be displayed on a application window using the **VC_DisplayCapture** command. This is usefull for quickly previewing the live capture and can also be used with the **VC_ColorCorrectImage** command to make color adjustments in real time. The **VC_SaveCaptureImage** can then be used to save the image or the **VC_DiscardPreviewImage** can be used release the image.

Params

Parameter 1 Window Handle that the image is displayed in. Be sure to cast this parameter as an integer
Parameter 2 X coordinate location of upper left corner of the image location.
Parameter 3 Y coordinate location of upper left corner of image location
Parameter 3 0

Example

VALCam_Control(VC_DisplayCapture, (int)hWND,20,10,0)
This command will display the captured image in window hWnd at column 20 row 10.

REGISTERYentries:

None

Notes:

It is the applications responsibility to make sure that the window handle used as **Param1** is valid.
This command will only return error free if an image was captured using **VC_LiveCapture** **VC_FlashCapture** with **Param 2 =1**. The command will return a negative value if there is not an image available for display.

Returns:

>0 success.
<1 indicates that a image was not available for display or the Window handle is invalid.

VC_ColorCorrectImage = 24

This command is used to color correct the captured image before it is saved. Red and Blue values can be subtracted or added to all pixels in the image. The last values used with this command are always applied to the capture image before saving. The last values used are saved to the **Registery**file as **Blue Paint** and **Red Paint**. These values are read and applied during the VC_InitSystem command.

This command can be used in conjunction with the **VC_DisplayCapture** command to color correct the image in real time See the sample applications for this usage.

Params

Parameter 1 Red Color Corretion value from -128 - 127.
Parameter 2 Blue Color Correction value from -128 – 127

REGISTERYentries:

Red Paint
Blue Paint

Notes

When used in conjunction with **VC_DisplayCapture**, this command can color correct the captured image in real time.

Returns

1 Sucess
<0 Failure

VC_SaveCapturedImage = 23

If an image is captured with **VC_LiveCapture** or **VC_FlashCapture** with **Param 2** =1, this command must be used to save the image. The image will be saved according to the value of **Param4** used in **VC_LiveCapture** and **VC_FlashCapture.** This command is used after the image has been previewed and color corrected.

If this command is not used then **VC_DiscardImage** must be before **VC_LiveCapture** or **VC_FlashCapture** can be used.

Params

Param 1, Param2, Param3, Param4 =0

REGISTERYentries:

None

Notes

VC_SaveCapturedImage or **VC_DiscardCaptured** must be called after **VC_LiveCapture** or **VC_FlashCapture** is called with **Param 2**= 1, before another **VC_LiveCapture** or **VC_FlashCapture** can be issued. If this command is called when a valid preview image does not exits it returns a value <1.

Returns

>1 Success

=0 Indicates an error for DIB and DDB captures. All other value are valid.

<1 Indicates failure for Clipboard, JPG and Clipboard

ERROR_SEQUENCE	**-1**	**Sequence error**
ERROR_NO_BITMAP	**-2**	**Invalid bitmap handle**
ERROR_MEMORY_TOO_LOW	**-3**	**Not enough memory available**
ERROR_FILE_LSEEK	**-4**	**Error seeking to position**
ERROR_FILE_WRITE	**-5**	**Error writing file**
ERROR_FILE_GONE	**-6**	**File not present - abort**
ERROR_FILE_READ	**-7**	**Error reading file**
ERROR_INV_FILENAME	**-8**	**Invalid filename specified**
ERROR_FILE_FORMAT	**-9**	**Invalid file format**
ERROR_FILENOTFOUND	**-10**	**File not found**
ERROR_INV_RANGE	**-11**	**Invalid width/height**
ERROR_IMAGE_TYPE	**-12**	**Image format recognized, but sub-type not supported**
ERROR_INV_PARAMETER	**-13**	**Invalid parameter passed**
ERROR_FILE_OPEN	**-14**	**Not able to open file**
ERROR_UNKNOWN_COMP	**-15**	**Unknown compression format**
ERROR_FEATURE_NOT_SUPPORTED	**-16**	**Feature not supported**
ERROR_NOT_256_COLOR	**-17**	**VGA card only supports 256 colors (8 bit)**
ERROR_PRINTER	**-18**	**Printer error**
ERROR_CRC_CHECK	**-19**	**Data CRC check error**
ERROR_QFACTOR	**-21**	**Invalid QFactor specified**
ERROR_TARGAINSTALL	**-22**	**TARGA not installed**
ERROR_OUTPUTTYPE	**-23**	**Invalid compression format**
ERROR_IMAGE_Exists	**-100**	**Previous image not deleted**

VC_DiscardCapturedImage = 25

After **VC_LiveCapture** or **VC_FlashCapture** is called with **Param 2** = 1 a temporary preview image is created. This image can be previewed (**VC_DisplayCapture**), and color corrected. The image must then either be saved (**VC_SaveCapturedImage**) or released before another capture command can be used..

VC_DiscardCapturedImage will release the image and free system resources

Params

Param1, Param2, Param3, Param4 =0

REGISTERYentries:

None

Notes

This command or **VC_SaveCaptured** must be called after **VC_LiveCapture** or **VC_FlashCapture** is called with **Param 2**= 1, before another **VC_LiveCapture** or **VC_FlashCapture** can be issued.

VC_MessageDrainHnd = 21

This command will allow a Window (or a MFC CWnd) object to respond to window messages on the live preview window. The sample application uses this command to allow the application to respond to mouse clicks on the live video to immediately center the subject in the display (PTZ version only)

See the Sample App for a demonstations.

Params

Param1 Window handle to receive messages.
Param 2-4 0

REGISTERYentries:

None

Set Flash Intensity

VC_SetFlashIrisLevel=5

This command sets the Flash intensity level. The flash intensity level will need to be adjusted for different subject distances to the camera**. Refer to the table in this doc for estimated settings for different subject distances**. Once this setting is adjusted it will not need to be changed unless the subject distance changes

Params

Param 1 is the Flash Intensity Level. Values from1 to 32 are valid. The higher the value the greater the flash intensity

REGISTERYentries:

FlashIrisLevel

Example

VALCam_Control(VC_SetFlashIrisLevel,14,0,0,0)

Returns

<1 indicates error conditions This command only set the size of the displayed image.

Notes

The Flash Intensity Setting is independent of room lighting and will only change when the subject distance changes from the camera. Higher settings increase the flash intensity and lower settings reduce the Flash intensity.

Refer to doc for suggested setting for different subject distances.

Unload Camera Drivers

VC_CloseCard=20

This command will unload the camera drivers from memory.
This command should only be issued when exiting the application or when the camera will not be needed for the duration of the application..

After this command is issued the camera will not respond until a **VC_InitSystem** command is sent.

After this command is issued the **REGISTERY**file will be updated.

Failure to call this command when exiting your application will cause an error condition.

NOTE:

During software development, if you exit or abort your application without calling VC_CloseCard, its recommened to unplug and plug the USB cable before restarting the DLL.

Params

Param 1-4 =-0

REGISTERYentries:

The Registery keys areupdated after this command is issued

Example:

VALCam_Control(VC_CloseCard,0,0,0,0);

Returns

<1 indicates an error condition.

Note

This command should only be used when exiting the application. Use VC_FreezeVideo command to turn off live video and release system resources. Use VC_NoDisplayVideo to disconnect the live video stream from a window.

Failure to use this command when exiting your application can cause an error condition.

Rotate Live Display

VC_RotateLiveDisplay = 250

This command will rotate the live video display in 90 degree increments

Params

Param1: This parameter can be 0, 90, 180, or 270.
Param2-4: 0

REGISTERYentries:

None

Example:

VALCam_Control(VC_RotateDisplay,270,0,0,0);

Returns

<1 indicates error conditions.

Notes

This command will rotate video in 90 degree to compensate for cameras that are rotated + or – 90 degrees. The **VC_DisplayVideo** and **VC_FlashCapture** commands will also provide video in the correct orientation.

The application must provide the correct parameters in the **VC_SetCropRectangle** and **VC_DisplayVideo** after this call or the live and captured video might be distorted. Refer to the sample app for an example.

`

Set Zoom Position

VC_SetZoomPosition = 303

This command will immediately set the camera zoom to any position accurately. A value of 0 is fully zoomed out. A value of 0X4000 is fully zoomed in.

Params

Param1: Value of the zoom position. See Notes for specific values at different zoom factors.

Example

VALCam_Control(VC_SetZoomPosition, 0X3EF7,0,0,0)

This will set the the zoom to a magnification of 16 x 1.

REGISTERYentries:

None

Notes

x1=0, x2=0X1606, x3=0X2151, x4=0X2860, x5=0X2CB5, x6=3060, x7=0X32d3, x8=0X3545, x9=0X3727, x10=0X38a9, x11=0X3a42, x12=0X3b4b,x13=0x3c85, x14=0X3d75, x15=0X3e4e, x16=0X3ef7, x17= 0X3fa0, x18=4000

Returns

<1 indicates error conditions This command only set the size of the displayed image.

NEW Set Zoom Array Position

VC_ZoomArray 230

This command will immediately set the camera zoom to preset setting between 0 - 49. Each increment increase zoom magnification by 2%. A value of 0 is least magnification (wide zoom) and a value of 49 is maximum magnification (zoomed in tight).

Params

Param1: Value of the zoom array position. Values can range from 0 - 49

Example

VALCam_Control(VC_ZoomArray, 25,0,0,0)

REGISTERYentries:

ZoomArray

Notes

Returns

<1 indicates error conditions

ReadZoom

VC_ReadZoom =300
This command returns the zoom position

Params 1-4 =0

Example:
int **ZoomPosition**
ZoomPosition= VALCam_Control(VC_ReadZoomPosition, 0X3EF7,0,0,0)

REGISTERYentries:
None

Notes
Reads current zoom position

Returns
Returns current zoom position.

Center Camera (Only available on PTZ model)

VC_SetPanTiltCenter = 305
This command will immediately center the camera

Params 1-4 =0

Example;
VALCam_Control(VC_SetPanTiltCenter, 0,0,0,0)

VALCam USB SDK.INI entries:
None

Notes
Only available on PTZ model

Returns
<1 indicates error conditions This command only set the size of the displayed image.

Set Pan/Tilt Pixel postion (Only available on PTZ model)

VC_SetPTZPosition = 304 //

This command will move the camera a set amount of pixels

Params

Param1: Amount of pixels to move the camera in the horizontal direction. A positive value will move the image to the right as viewed in the monitor. A negative value will move the image to the left.

Param2: Amount of lines to move the camera in the vertical direction. A positive value will move the image up as view in the monitor. A negative value will move the image down a viewed in the monitor.

Example;

VALCam_Control(VC_SetPTZPosition -234,6,0,0)

This command will move the camera 234 pixels to the left (as viewed in the monitor) and 6 pixels up (as viewed in the monitor)

VALCam USB SDK.INI entries:

None

Notes

This command is only available on PTZ model.

Returns

<1 indicates error conditions.

Camera Zoom Controls Commands

VC_ZoomIn = 13, VC_ZoomInFast= 228, VC_ZoomInSlow =222
VC_ZoomOut = 14, VC_ZoomOutFast = 229, VC_ZoomOutSlow = 223

VC_ZoomStop = 14

Params
Param 1-4 -0

Example
VALCam_Control(VC_ZoomOutFast,0,0,0,0)
Sleep(500);
VALCam_Control(VC_ZoomStop,0,0,0,0)

Camera will zoom out fast for 500ms and then stop.

REGISTERYentries:
None

Returns
<1 indicates error conditions This command only set the size of the displayed image.

Live Brightness

VC_LIvelrisUp = 42
This command will incrementally increase the live brightness level

VC_LivelrisDown = 43
This command will incrementally decrease the live brightness level

Params 1-4 =0

REGISTERYentries:
None

Notes
This command only effects the live preview level and live captures. The brightness of the Flash Captures is not effected. See **VC_FlashCapture** and **VC_FlashIrisLevel**

Returns
<1 indicates error conditions This command only set the size of the displayed image.

NEW Set Live Brightness

VC_SetExposure 132
This command will incrementally increase the live brightness level

Params 1
This parameter sets the exposure level. This value can range from 1 -15.

REGISTERYentries:
None

Example
VALCam_Control(VC_SetExposure,7,0,0,0)

Notes
This command only effects the live preview level and live captures. The brightness of the Flash Captures is not effected. See **VC_FlashCapture** and **VC_FlashIrisLevel**

Returns
<1 indicates error conditions This command only set the size of the displayed image.

NEW Return Live Exposure

VC_ReadExposure =133

This command will return the current exposure setting. These values range from 1 15.

Params 1 -4

0

Example

int ExposureLevel = VALCam_Control(VC_SetExposure,7,0,0,0)

REGISTERYentries:

None

Notes

This command only effects the live preview level and live captures. The brightness of the Flash Captures is not effected. See **VC_FlashCapture** and **VC_FlashIrisLevel**

Returns

<1 indicates error conditions

Live Color Settings

VC_SetRedLIve=110

This command sets the Red gain to the value of **Param1** in the live preview and live capture. Valid values are from -50 to 50. Default is 0.
Adjusting this command turns off **AutoWhite** mode.

REGISTERYentries:

RedPreviewOffset = Param1
AutoWhite=0

VC_SetBlueLIve=111

This command sets the Blue gain to the Value of **Param1** in the live preview and live capture. Valid values are from -50 to 50. Default is 0.
Adjusting this command turns off **AutoWhite** mode.

REGISTERYentries:

BluePreviewOffset= Param1
AutoWhite=0

VC_SetColorLive=112

This command sets the Color intensity to the value in **Param1** in the live preview and live capture. Valid values are from 0 to 110. Default is 69. Higher values increase color intensity, lower values decrease color intensity.

REGISTERYentries:

ChromaLevel= Param1

Param1

Param1 sets the value of the Red gain, Blue gain or Color intensity. Valid values are from -50 to 50 for Red and Blue with a default of 0. Color intensity can range from 50 to 110 with a default of 69.

Example

VALCam_Control(VC_SetRedLIve, -10,0,0,0) // Decreases Red Gain to -10
VALCam_Control(VC_SetBlueLIve, 23,0,0,0) // **In**creases Red Gain to 23
VALCam_Control(VC_SetColorLIve, 89,0,0,0) // **In**creases Color Gain to 89

NOTE:

These commands only effect the live color settings. These setting do not affect flash captures.

Flash Color Settings

VC_SetRedFlash=113

This command sets the Red gain to the value of **Param1** during the Flash capture. **Please note that the effects of this command will only appear after the Flash is triggered when viewing the captured image**. Valid values are from 50 to 50. **Default is 0**

REGISTERYentries:

RedFlashOffset= Param1

VC_SetBlueFlash=114

This command sets the Blue gain to the value of Param1 during the Flash capture. **Please note that the effects of this command will only appear after the Flash is triggered when viewing the captured image**. Valid values are from 50 to 50. **Default is 0**

REGISTERYentries:

BlueFlashOffset= Param1

VC_SetColorFlash=115

This command sets the Color intensity to the value of Param1 in the Flash capture. **Please note that the effects of this command will only appear after the Flash is triggered when viewing the captured image**. Valid values are from 0 to 110. **Default is 80**

REGISTERYentries:

FlashChromaLevel= Param1

Param1

Param1 sets the value of the Red gain, Blue gain or Color intensity. Valid values are from -25 to 25 for Red and Blue with a default of 0. Color intensity can range from 20 to 110 with a default of 60. **NOTE: These changes only appear after the Flash capture when viewing the captures image**

Example

VALCam_Control(VC_SetRedLIve, -10,0,0,0) // Decreases Red Gain after Flash to -10
VALCam_Control(VC_SetBlueLIve, 23,0,0,0) // Increases Red Gain after Flash to 23
VALCam_Control(VC_SetColorLIve, 89,0,0,0) // Increases Color Gain after Flash to 89

Notes

These commands are **Extremely** important for properly adjusting flesh tones and color intensity. Its highly recommend that the developer experiment with using these commands in the sample program to understand their usage.
NOTE: These changes only appear after the Flash capture when viewing the capture image

Set Live Exposure

VC_SetExposure = 132

This command will continually increase the live brightness level. This command can be used to adjust live brightness via a scroll bar type control

Params 1=0 - 15. Higher values display brighter live preview

Example

VALCam_Control(VC_SetExposure,7,0,0,0,)

Sets live brightness to a level of 7 (out of 15)

REGISTERYentries:

None

Returns

<1 indicates error conditions This command only set the size of the displayed image.

Notes

This command only effects the live preview level and will not adjust the brightness of the Flash Captures.

Set Flash Color Correction

VC_SetFlashColorCorrection = 119
Constant=119

Param1
0- No Correction
1 - Correction of Fluorescent Lighting

Registeryentries:
FlashColorCorrectionType

Notes
This function, when passed with a Parameter of 1 will offset the FlashRed and FlashBlue controls to automatically adjust for Fluorescent lighting. The RedFlash and BlueFlash controls are still available for adjustment, but their center positions will be offset for fluorescent lighting. A value of 0 should be passed to this function when operation under daylight lighting (including daylight fluorescents).

Reading Status Information

VC_ReturnFlashIris = 39
VC_ReturnLiveRed = 31
VC_ReturnLiveBlue = 32
VC_ReturnLiveColor = 33
VC_ReturnFlashRed = 34
VC_ReturnFlashBlue = 35
VC_ReturnFlashColor = 36
VC_ReturnFGSharpness = 131
VC_ReturnSharpness = 130
VC_ReturnCropWidth = 154
VC_ReturnCropHeight = 155
VC_ReturnCropStartX=152
VC_ReturnCropStartX=153
VC_ReturnContrast = 50
VC_ReturnBrightness =51
VC_ReturnFlashColorCorrectionType = 157

Params 1-4 = 0

Example
FlashIrisLevel = VALCam_Control(VC_ReturnFlashIris,0,0,0,0)

NEW Reading State Parameters with One Call

VALCam_ReadReturnParameters(ptrstructReturnValues ReadReturnParameters)

This exported DLL function call will return all return values in a pointer to a structure passed from the application.

```
struct structReturnValues
{
      int ReturnLiveRed;
      int ReturnLiveBlue;
      int ReturnFlashRed;
      int ReturnFlashBlue;
      int ReturnFlashIris;
      int ReturnSharpness;
      int ReturnFGSharpness;
      int ReturnAutoWhite;
      int ReturnRedPaint;
      int ReturnBluePaint;
      int ReturnCropStartX;
      int ReturnCropStartY;
      int ReturnCropHeight;
      int ReturnCropWidth;
      int ReturnFlashCorrection;
      int ReturnZoomArray;
      int ReturnLiveIrisLevel;
      int ReturnPTZSpeed;
      int ReturnReserved2;
      int ReturnReserved3;
      int ReturnReserved4;
      int ReturnReserved5;
      int ReturnReserved6;
      int ReturnReserved7;
};
typedef structReturnValues *ptrstructReturnValues;
```

NOTE:

The application must create the structure and pass a pointer to the structure in the call. The individual **VC_RetrunXXX** calls can be used to return these values separately.

Set Live Preview Defaults

VC_SetPreviewDefaults = 27

This command performs the following functions:

1. White Balances the camera using older white balance call
2. Set Live Red Gain = **DefaultRedPreviewOffset ;**(located in INI, =0)
3. Set Live Blue Gain =**DefaultBluePreviewOffset;** (located in INI, =0)
4. //Set Live Color Level = **DefaultChromaLevel;** (located in INI,=69)
5. //Set Contrast= **DefaultContras**t; (located in INI,=64)
6. //Set Brightness=**DefaultBrightness**; (located in INI,=124)

Params 1 -4 = 0

Example

VALCam_Control(VC_SetPreviewDefaults,0,0,0,0)

Notes

New Default values can be set in the **REGISTERY**file with these entries:
DefaultRedPreviewOffset. DefaultBluePreviewOffset, DefaultChromaLevel DefaultContrast, DefaultBrightness, DefaultShaprness, DefaultAperature

Set Flash Preview Defaults

VC_SetFlashDefaults = 28

This command performs the following functions:

1. Set Flash Red Gain = **DefaultRedFlashOffset ;**(located in INI, =0)
2. Set Flash Blue Gain = **DefaultBluePreviewOffset;** (located in INI, =0)
3. Set Flash Color Level = **DefaultFlashChromaLevel;** (located in INI,=80)
4. //Set Contrast= **DefaultContrast;** (located in INI,64). NA
5. //Set Brightness= **DefaultBrightness;** (located in INI,124). NA

Params 1 -4 = 0

Example

VALCam_Control(VC_SetFlashDefaults,0,0,0,0)

Notes

New Default values can be set in the **REGISTERY**file with these entries:
DefaultRedFlashOffset. DefaultBlueFlashOffset, DefaultFlashChromaLevel DefaultContrast, DefaultBrightness, DefaultSharpness, DefaultAperature.

Set Frame Grabber Sharpness
VC_FGSharpness = 120

This command sets the Frame Grabber sharpness from a value of 0 – 4. Higher values result in a sharper image but if the value is too high the image will become edgy and grainy. The default value is 3.

Param1

Set to value of **FGSharpness** of 0 - 4

VALCam USB SDK.INI entries:

Aperature

Example

VALCam_Control(VC_FGSharpness, 3, 0, 0, 0)

NEW Capture Exported Function:

Usage:

VALCam_Capture(prtstructCapture CaptureStruct)

The application passes a pointer to the following structure:

```
struct structCaptureConfiguration
{
int Sharpness;          //Shaprness;  //values 0 - 900, 600 default if 0
int Preview;            //Preview;
char * FileName;        //FileName; //Full path and file name;
int FileType;           //0-DIB, 1-DDB, 2-Clipboard, 3-BMP, 4-JPG
int JpgCompression;     // 1-Min Compression, 255-Max Compression
int FlashOn;            //0-Flash Off, 1- Flash On
};
typedef structCaptureConfiguration *prtstructCapture;
```

Example:

```
structCapture. Shaprness=500;
structCapture.Preview=0;
structCapture.FileName="Test.JPG";
structCapture.FileType=4;
structCapture.JpgCompression=25;
structCapture.FlashOn=1;

VALCam_Capture(&structCapture);
```

Return Values:

Refer to VC_LiveCapture and VC_Flash Capture.

NOTE:

This call is functionally equivalent to **VC_LiveDisplay** and **VC_FlashDisplay** except that a file can be uniquely named. Using **VC_LiveDisplay** and **VC_FlashDisplay JPG** and BMP files are always named**: Image4.JPG / Image4.BMP**

NEW Capture Buffer exported Function

Usage:

int **VALCam_CaptureToBuffer(unsigned char * ImageBuffer, int Preview, int Flash)**

The application will pass a pointer to buffer (array) of bytes. The call will fill the buffer with the raw image data of 24 bits/pixel. The buffer must be equal to **CropWidth** X **CropHeight** X 3.

Return:

This call will return the size of the buffer with raw data.

NOTE:

The buffer size must = size of the raw captured data.
This size will be **CropWidth** X **CropHeight** X 3.
These values are set with the **VC_SetCropParameters** command
CropWidth and **CropHeight** are stored in the Registery and can be obtained the **VC_ReturnCropHeight**, **VC_ReturnCropWidth** and the **VALCamReturnParameters** call.

.

Flash iris setting and estimated subject distances.

The following chart estimates a given flash iris setting and subject distance from the camera.

Please note that these are estimates only and the correct value might vary plus or minus 1 or 2 settings. Test captures should be taken on initial installation to optimize this value.

Once the optimal setting is found its value will not change unless the subject distance from the camera changes. Values for both the new **camera,type 5**, and old camera, **type 4** are provided.

Flash Iris Setting	New Camera(type 5)	Old camera (type 4)
1	3 in	Under 1.2 feet
2	4 in	Under 1.2 feet
3	5 in	Under 1.2 feet
4	6 in	Under 1.2 feet
5	7 in	Under 1.2 feet
6	1 ft	Under 1.2 feet
7	1.1 ft	1.5 feet
8	1.2 ft	1.7 feet
9	1.3 ft	1.8 feet
10	1.4 ft	2 feet
11	1.6 ft	2,2 feet
12	1.8 ft	2.3 feet
13	2 ft	2.4 feet
14	3 ft	2.5 feet
15	4 ft	2.6 feet
16	5 ft	2.7 feet
17	6 ft	2.7 feet
18	7 ft	2.8 feet
19	8 ft	2.9 feet
20	9 ft	3 feet
21	10 ft	3.3 feet
22	11 ft	3.8 feet
23	12 ft	4 feet
24	13 ft	4.5 feet
25	14 ft	5 feet
26	15 ft	5.5 feet
27	16 ft	6 feet
28	16.5 ft	6.5 feet
29	16.8	7 feet
30	17.5 ft	8 feet
31	16 ft	9 feet
32	20 ft	10 feet

Registery Settings

All state parameters are save in the registery.
These values are located in:

HKEY_CURRENT_USER,\Software\ VALCam USB SDK Zoom

The DLL writes these values when the **VC_CloseCard** command is issued.
The DLL reads these values when the **VC_InitSystem** command is issued.

The individual setting can be read by issuing **VC_ReturnXXX** for the specific value
All values can be read at once in a structure by calling:

VALCam_ReadReturnParameters(ptrstructReturnValues ReadReturnParameters)

Hints and Tips

Use the VC_InitSystem command only once in your application.

The **VC_InitSystem** command takes a couple of seconds to complete. You only need to issue this command one time in your application. If you need to display the Live Video in another window handle, use the **VC_SetWindowHandle** command.

VC_InitSystem should not be called again until VC_CloseCard has been issued.

Use the VC_FreezeVideo to freeze video and to release system resources when the camera. is not needed. The **VC_FreezeVideo** command should be issued when the Live Video Window is not needed. This will release system resources that would normally be used. Issue the **VC_DisplayVideo** command to display live video. **VC_LiveCapture** and **VC_FlashCapture** can be issued anytime.

Issue the VC_Close command when exiting your application

VC_Close must be issued when terminating your application to release **DirectX** resources. Failure to issue this command will cause an error when trying to use **VC_InitSystem** in a new session

Zoom in as tight to subject as image composition will allow.

Zooming in as close to subject as image composition will allow will maximize image quality. Do not zoom our and crop in software as this will degrade image resolution.

Adjust the Flash Iris Settings for optimal flash illumination.

The Flash Iris setting is a simple adjustment that can make all the difference in image quality when using the flash. Once this control is set properly, changes in room lighting will not affect image quality. You can even turn off all lights in the room and capture identical images with no additional adjustments. The effects of room lighting is almost totally eliminated

Experiment with Red, Blue and Color controls for both Flash and Live settings

Although the default settings will be close to optimal for most cases, it is recommend that the **Red, Blue** and **Color** controls are adjusted to get a feel of their capabilities. These controls have a fine granularity and can fine tune color, flesh tone and background issues to suit most situations. These software default settings can be changed in the "**DCS 8000 SDK.INI"** file if desired.

Experiement with the Sharpness and Aperature controls

The **Sharpness** and **Aperature** controls offer a range of setting from a "soft", digital camera look to a sharp edgy look that may be more suited for printed photo-ids. Different printers may work better with different Sharpness settings. As the sharpness setting is increased noise in the capture is also increased.

SDK Definitions

Command Constants:

```
#define VC_InitSystem 0
#define VC_DisplayVideo 1
#define VC_Freeze   2
#define VC_NoVideoDisplay 3
#define VC_LiveCapture 4
#define VC_SetFlashIrisLevel 5
#define VC_SendWindowHandle  6
#define VC_FlashCapture 7

#define VC_TiltUp   8
#define VC_TiltDown 9
#define VC_PanLeft 10
#define VC_PanRight 11
#define VC_PanTiltStop 12
#define VC_ZoomIn 13
#define VC_ZoomOut 14
#define VC_ZoomStop 15

#define VC_UnloadDriver 20
#define VC_MessageDrainHnd 21
#define VC_DisplayCapture  22
#define VC_SaveCapturedImage     23
#define VC_ColorCorrectImage 24
#define VC_DiscardCapturedImage 25
#define VC_TiltUpSlow   218
#define VC_TiltDownSlow 219
#define VC_PanLeftSlow  220
#define VC_PanRightSlow 221
#define VC_ZoomInSlow   222
#define VC_ZoomOutSlow  223
#define VC_TiltUpFast   224
#define VC_TiltDownFast 225
#define VC_PanLeftFast  226
#define VC_PanRightFast 227
#define VC_ZoomInFast   228
#define VC_ZoomOutFast  229

#define VC_RotateLiveDisplay 250

#define VC_SaveIniFile  17
#define VC_CloseCard  20

#define VC_LiveIrisUp 42
#define VC_LiveIrisDown 43

#define VC_SetPreviewDefaults 27
#define VC_SetFlashDefaults 28

#define VC_ReturnLiveRed 31
#define VC_ReturnLiveBlue 32
#define VC_ReturnLiveColor 33
#define VC_ReturnFlashRed 34
#define VC_ReturnFlashBlue 35
#define VC_ReturnFlashColor 36
```

```
#define VC_ReturnFlashIris 39
#define VC_ReturnContrast   50
#define VC_ReturnBrightness 51
#define VC_ReturnSharpness  130
#define VC_ReturnFGSharpness 131
#define VC_ReturnCameraSharpness 131
#define VC_ReturnAutoWhite 134
#define VC_ReturnRedPaint 135
#define VC_ReturnBluePaint 136
#define VC_SetExposure 132    //Set Brightness level 0-14, Flash
                              //Captures are not affected
#define VC_ReadExposure 133
#define VC_SetBrightness 116
#define VC_SetContrast   117
#define VC_SetFlashColorCorrection 119
#define VC_ReturnCropStartX 152
#define VC_ReturnCropStartY 153
#define VC_ReturnCropWidth 154
#define VC_ReturnCropHeight 155
#define VC_ReturnFlashColorCorrectionType 157

#define VC_SetCropParameters 156
#define VC_ReturnZoomArray 158
#define VC_ReturnExposureComp 159

#define VC_AutoWhite  118
#define VC_SetRedLive 110
#define VC_SetBlueLive 111
#define VC_SetColorLive 112
#define VC_SetRedFlash 113
#define VC_SetBlueFlash 114
#define VC_SetColorFlash 115

#define VC_SetFrameGrabberSharpness 120
#define VC_FastCapture 200

#define VC_ZoomArray   230
#define VC_ReadZoomPosition 300
#define VC_SetZoomPosition  303
#define VC_SetPanTiltPosition 304
#define VC_SetPanTiltCenter 305
#define VC_AccurateAutoWhite 125
#define VC_VerifySystemInitialized 600
```

Sturction Definitions:

```
struct structReturnParameters
{
	int RedPreviewOffset;
	int BluePreviewOffset;
	int RedFlashOffset;
	int BlueFlashOffset;
	int FlashIrisLevel;
	int ImageSharpness;
	int CameraSharpness;
	int AutoWhite;
	int RedPaint;
	int BluePaint;
	int CropStartX;
	int CropStartY;
	int CropHeight;
	int CropWidth;
	int FlashCorrectionType;
	int ZoomArray;
	int ExposureComp;
	int PTZSpeed;
	int Reserved2;
	int Reserved3;
	int Reserved4;
	int Reserved5;
	int Reserved6;
	int Reserved7;
};
typedef structReturnParameters *pReturnParameters;

struct structCaptureParameters
{
	int Sharpness;
	int PreviewCapture;
	CString FileName;
	int CaptureType;
	int JPGCompression;
	int FlashOn;

typedef structCaptureParameters *pCaptureParameters;
};
```

Exported Function Definitions

```
extern "C" _declspec(dllexport) int CALLBACK VALCam_Control(int
Message,int Param1,int Param2, int Param3, int Param4);

extern "C" _declspec(dllexport) int CALLBACK
VALCam_CaptureToBuffer(unsigned char * ImageBuffer, int Preview, int
Flash );

extern "C" _declspec(dllexport) int CALLBACK
VALCam_ReadReturnParameters(structReturnParameters* pReturnParameters);

extern "C" _declspec(dllexport) int CALLBACK
VALCam_Capture(structCaptureParameters *pCaptureParameters);

extern "C" _declspec(dllexport) int CALLBACK
VALCam_CaptureToPreviewBuffer(unsigned char * ImageBuffer);
```