



UDA5 VCA5

SDK Manual

Contents

What's new	4
1. About VCA5 API	6
1.1. Features	8
1.2. Minimum System Requirements	10
1.3. Package Contents.....	12
1.4. Terminology	12
2. Using VCA5 API	15
2.1. Process state	15
3. VCA5 API Reference	17
3.1. Functions.....	17
VCA5Init	17
VCA5GetLastErrorCode	18
VCA5QueryInfo	19
VCA5Activate	21
VCA5GetSystemInfo.....	22
VCA5GetEngineInfo.....	23
VCA5Open	25
VCA5Close	26
VCA5Process	27
VCA5Control.....	29
3.2. Structures.....	33
VCA5_HWGUID_INFO	33
VCA5_LICENSE_INFO.....	34
VCA5_SYSTEM_INFO	36
VCA5_ENGINE_INFO	37
VCA5_ENGINE_PARAMS	39
VCA5_ZONE.....	40
VCA5_RULE	41
VCA5_RULE_LINECOUNTER	44
VCA5_RULE_TIMER	45
VCA5_RULE_COLSIG.....	45
VCA5_SUBCOUNTER	46
VCA5_COUNTER.....	47
VCA5_CALIB_INFO	48

VCA5_CLSOBJECT	50
VCA5_TAMPER_INFO	51
VCA5_XMLCFG_PARAMS	52
VCA5_TRACKERPARAMS	53
VCA5_SCENECHANGE_INFO	54
3.3. Rule type	55
3.4. Camera calibration and object classification	56
3.4.1. Camera calibration	56
3.4.2. Object classification	56
Revision History	57

What's new

V1.0.3

[ADD] Support camera calibration and object classification.

[ADD] Additional VCA5Control commands

- VCA5_CMD_SETCALIBPARAMS
- VCA5_CMD_SETOBJECT
- VCA5_CMD_CLEAROBJECT
- VCA5_CMD_SETSTABENABLE
- VCA5_CMD_SETRETRIGTIME

[CHG] VCA5_SYSTEM_INFO structure

[CHG] VCA5_ENGINE_INFO structure

[CHG] VCA5_RULE structure

[ADD] VCA5_CALIB_INFO structure

[ADD] VCA5_CLSOBJECT structure

[ADD] Speed filter - VCA5_RULE_TYPE_SPEED

V1.0.4

[CHG] Allow variable FPS and resolutions.

V1.0.5

[ADD] Support ECPRR480D-16EX

[ADD] Support Windows 7

V1.0.6

[ADD] Support ECP240 Series

[ADD] Support ECP960-32EX

V1.0.7

[ADD] VCAsysPCOpen Package

[ADD] Tamper detection

V1.0.8

[ADD] Support NCP Series

[ADD] Support HP4000EX Series

V1.0.9

[ADD] Load/save XML configuration

V1.2.0

[ADD] VCAcount, VCAadvanced license

[ADD] VCA5Init function

[CHG] VCA5GetEngineInfo

[CHG] VCA5_LICENSE_INFO

[CHG] VCA5_ENGINE_PARAM

[CHG] VCA5_RULE

[ADD] VCA5_RULE_LINECOUNTER

[ADD] VCA5_RULE_TIMER

[ADD] VCA5_CLSOBJECT
[ADD] VCA5_TRACKERPARAMS
[ADD] Counting line filter, tailgate filter
[CHG] Support Image size

V1.2.1

[ADD] Abandoned/Removed Object filter
[ADD] Stationary blob
[CHG] VCA5_TRACKERPARAMS
[CHG] Engine run based not real-time clock but input timestamp.

V1.3.0

[ADD] Fire/Smoke filter
[ADD] Fire/Smoke blob
[CHG] VCA5_TRACKERPARAMS
[CHG] Support License count Max 64.
[CHG] Change Data type in header file

V1.4.0

[ADD] Certification VCA5Lib
[CHG] Support Fire&Smoke in Advanced and Surveillance license
[CHG] VCA5_TRACKERPARAMS

V1.4.1

[ADD] SceneChange Detection filter
[ADD] IR-Cut Detection in SceneChange Detection filter
[ADD] VCA5_SCENECCHANGE_INFO

V1.4.2

[ADD] Color filter
[ADD] People tracker engine
[ADD] VCA5_RULE_COLSIG
[ADD] VCApro license
[ADD] 64bit Core support

1. About VCA5 API

VCAsys is a real-time video analytics engine that utilizes advanced image processing algorithms to turn video into actionable intelligence. At the core of the product, there is an advanced object tracking engine that continually tracks moving and stationary targets. The tracking engine features built-in robustness to environmental nuisance conditions such as changing illumination, moving foliage, rippling water, etc.

There are 6 levels of functionality:

VCApresence: continually tracks moving and stationary targets and generates real-time alerts of object presence in multiple overlapping detection zones.

VCAsurveillance: continually tracks and classifies moving and stationary targets and features a full suite of rule-based filters including: enter, exit, appear, disappear, stopped objects, directionality constraints, object counting, loitering, object type, object speed and fire/smoke detect. Multiple filters are supported on any combination of multiple overlapping detection zones.

VCApro: includes VCAsurveillance features, and supports people tracker and color filter enabling an advanced people tracking engine optimized for tracking people in cluttered indoor scenes such as retail scenarios and specific high-accuracy counting functions optimized for use in busy scenes.

VCAcount: Bi-directional counting in busy areas.

VCAfire/smoke: Support Fire/smoke filter.

VCAadvanced: Combined VCAcount, VCAsurveillance and VCAfire/smoke.

There are 2 types of license available:

VCAsysPC (*VCApresPC* & *VCAsurvPC*): run on a PC platform and are integrated with our all range of NCP, ECP, ECPR, MP and HP series video capture cards.

VCAsysPCopen (*VCApresPCopen* & *VCAsurvPCopen*): also runs on a PC platform but independently of there being any PCVCA hardware in the system and hence can be used with any type of PC or IP capture device from other manufacturers.

Figure1. VCAsysPC

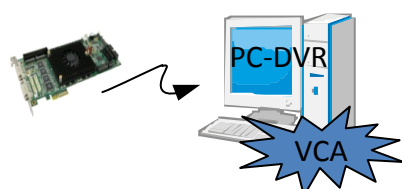
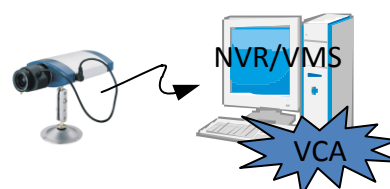


Figure2. VCAsysPCopen



VCA5 API is an API set designed to control the PC VCA library (VCAsysPC) and process a captured image from capture cards via UDA5 APIs. UDA5 refers to API version 5 among the various APIs

we provide. UDA5 consists of Cap5, Aud5, Cod5, Ovr5, Net5 and VCA5 API sets. For further information concerning each API set except for VCA5 API, refer to the SDK Manual for UDA5.

1.1. Features

GENERAL FEATURES

- High performance, comparable with the best in the industry.
- Quick and easy to use – intuitive graphical interface (also no algorithm parameters to confuse users).
- Realistic OEM prices.
- Multi license support
- Engine runs by using input timestamp of the moment which means created time of video.
- Enable/Disable engine (object tracking, stabilizer, counting line, Abandoned/removed object detection, Fire/Smoke)

TRACKER

- Tracks up to 100 targets simultaneously.
- Up to 40 general purpose, user-defined, polygonal detection zones.
- Camera shake cancellation – tracker works even though the camera is on a swaying pole.
- Tracks objects when they stop or move very slowly.
- Continues to track objects which pass each other, ensuring that the trail is continuous and unbroken.
- Continues tracking when objects are partially or completely obscured temporarily.
- Ignores changes in light due to clouds and switching on artificial lights or camera auto-iris operation.
- Ignores repetitive movement – waving trees, rippling water.
- Ignores cloud shadows passing across the ground.
- Ignores video interference and missing video frames.
- Adjusts to image degradation caused by rain, fog, dirty camera lens and low sun position causing dazzle.
- Does not false alarm due to PTZ camera movement.
- Tracks objects on 100% of screen area (except in any non-detection zones defined by the user).
- Includes on-screen Annotation Metadata – draws drawing bounding boxes and object trails.
- Highlights alarmed and non-alarmed objects in different colours.

VCApresence FEATURES

- Includes VCAsys tracker
- The Presence filter detects when an object such as a person or vehicle is inside or is crossing a zone or a line.
- Camera attack detection: detects moving, defocusing or covering of the camera. It does not respond to slow changes such as build-up of dirt on the camera lens or natural variations in daylight levels. (Available in the next release).
- Polygonal detection zones with a variable number of vertices so that any shape can be supported.
- Detection lines also supported, which can have one or many segments.
- Includes non-detection zones for masking possible sources of false alarms.
- Fast moving objects are detected even if they pass through a zone or line quickly.
- Camera tamper detection

VCAsurveillance FEATURES

- Includes all features of VCApresence.
- Up to 40 on-screen counters that can be linked to detection rules.
- Direction filter with independent adjustment of direction and acceptance angle.
- Stopping filter with variable time threshold.
- Dwell filter with variable time threshold, used to detect loitering people, etc.

- Enter and Exit filters.
- Appear and Disappear filters.
- Unique, easy to use, 3D calibration suitable for overhead and side-viewing cameras. Calibration in metric or imperial units.
- On-screen display of object classification, speed, area and height.
- Speed filter with upper and lower speed thresholds.
- Filter on object class – for example people, group of people, vehicle, clutter, etc.
- Filters out small animals, birds and blowing trash by excluding objects in the clutter category.
- Object classes can be changed by the user.
- Combined filter using object class and one other filter.
- Includes full Forensic Metadata stream in XML format for customer use.
- Abandoned /removed object
- Support stationary blob

VCApro FEATURES

- Includes all features of VCAsurveillance.
- People tracker
- Color filter

VCAcount FEATURES

- Counting line – optimized algorithm specifically for counting humans and vehicles, even in busy scenes.

VCASmokeFire FEATURES

- Fire detect – optimized algorithm specifically for find fire.
- Smoke detect - optimized algorithm specifically for find Smoke.

VCAadvanced FEATURES

A hybrid license that includes:

- all features of VCAcount.
- all features of VCAsurveillance.

Comparison between VCApresence and VCAsurveillance

	VCApresence	VCAsurveillance	VCAcount	VCAadvanced	VCASmoke Fire	VCAPro
Presence filter	√	√		√		√
Camera shake cancellation	√	√		√		√
Enter/exit filter		√		√		√
Appear/disappear filter		√		√		√
Stopped filter		√		√		√
Dwell filter		√		√		√
Direction filter		√		√		√
Speed filter		√		√		√
Counting		√	√	√		√
Calibration/classification		√		√		√

Metadata	√	√		√		√
Tamper detection	√	√		√		√
SceneChange detection	√	√		√		√
Tailgating filter		√		√		√
Counting line			√	√		√
Abandoned /removed		√		√		√
Smoke filter		√		√	√	√
Fire filter		√		√	√	√
Color filter						√



VCApresence encrypts the meta data of one object which is selected randomly. To draw the meta data properly, use the VCAMetaRender library. For more detailed information about VCAMetaRender library, please refer to VCA MetaRender API Manual.pdf.



The multiple licenses per each engine are available since SDK V1.2. VCACount license and VCAsurveillance license can be assigned to an engine at the same time.

1.2. Minimum System Requirements

HW Requirement

- Pentium Celeron or higher.
- Capture card manufactured by PCVCA Technology, if not using VCAsysPCOPEN

SW Requirement

- DirectX 7.0 or later (8.0 or later recommended).
- MS XML 6.0 or higher
(<http://www.microsoft.com/download/en/details.aspx?id=3988>)
- Microsoft Windows 2000, Windows XP, Windows Vista, Windows 7, Windows8
Capture card driver

NCP Series

	DLL	SYS
All NCP series	CAPSERIES.DLL V3263 or higher version	CAPSV.SYS V3270 or higher version

ECP Series

	DLL	SYS
ECP 240 Series	ECPSVD.DLL V2000 or higher version	ECPSVD.SYS V2000 or higher version
ECP960-32EX	ECPSVC.DLL V2000 or higher version	ECPSVC.SYS V2000 or higher version

ECPR Series

	DLL	SYS
ECPR480 Series ECPR240 Series ECPR120 Series	ECPSV.DLL V1802 or higher version	ECPSV.SYS V1802 or higher version
ECPR480D-16EX	ECPSV.DLL V2000 or higher version	ECPSVU.SYS V2000 or higher version

MP Series

	DLL	SYS
MP2000 MP3000(EX) BMP4000EX	ECPSV.DLL V2004 or higher version MPJVS.DLL V 2010 or higher version	ECPSV.SYS V2010 or higher version MPJVS.SYS V2010 or higher version
MP4000L	ECPSV.DLL V2004 or higher version MPJVS.DLL V 2010 or higher version	ECPSVL.SYS V2000 or higher version MPJVSL.SYS V2000 or higher version

HP Series

	DLL	SYS
HP2000 HP3000(EX) BHP4000EX	HPSV.DLL V2000 or higher version	HPSV.SYS V2000 or higher version
HP4000EX	HP4K.DLL V1.7.6.0 or higher version	HP4K.SYS V5.3.0.2 or higher version

HM Series

	DLL	SYS
HM4000EX BHM4000EX	HMSV.DLL V2.1.7 or higher version	HMSV.SYS V2.1.2 or higher version

NOTE: HM series do not support 64bit windows system.

1.3. Package Contents

Library

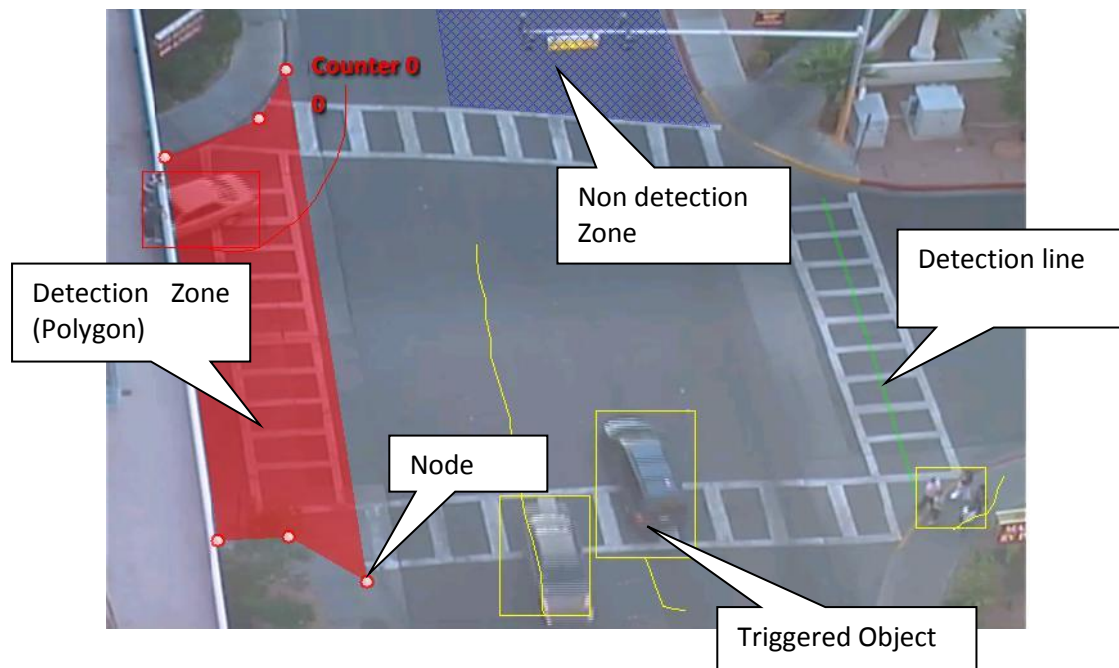
- VCA5CoreLib.h, VCA5Types.h
- VCA5Lib.lib
- VCA5Lib.dll

Example Applications

- | | |
|-------------|--|
| • DemoVCA | Demonstration application that illustrates the whole range of VCA functionality. |
| • SimpleVCA | Simple win32 program that illustrates the basics of the VCA5 control and metadata parsing and drawing. |
| • StartVCA | Simple console program that illustrates the basics of the initialization and processing sequence of the API. |
| • HWGUIDGen | Simple console program that generate GUID. |

1.4. Terminology

- | | |
|-----------------|--|
| • Zone: | A polygonal detection area that indicates an area in which a detection rule may operate. |
| • Object: | Any moving or stationary target that is being tracked by the tracking engine. |
| • Line: | A multi-point line that indicates a line over which a detection rule may operate. |
| • Rule: | A condition, or set of conditions, that is analyzed against a list of objects to determine whether a particular detection scenario has occurred. |
| • Event: | A notification that a detection rule has been triggered. |
| • Counter: | Maintains multiple or individual counts of events that have been raised by detection rules, provided by Sub Counters. |
| • Sub Counter: | An individual source of counting events generated by detection rules. |
| • Counting line | A special rule optimized for counting humans or vehicles, even in busy environments. |



1.5. Specification

Supported capture card models

ECP Series	ECP960-32EX ECP240-32, ECP240-32EX ECP240-16 ECP240-8
ECPR Series	ECPR480D-16EX ECPR480-16EX, ECPR480-16, ECPR480-16L ECPR240-16EX, ECPR240-16, ECPR240-16L ECPR120-16EX, ECPR120-16, ECPR120-16L ECPR120-4
MP Series	MP3000EX, MP3000 MP2000 MP4000L
NCP Series	NCP100, NCP1200, NCP2000, NCP3000, NCP3100, NCP3200
HP4000EX Series	HP4000EX
HM Series	HM4000EX, BHM4000EX

NOTE: HM series do not support 64bit windows system.

Supported max channels of VCApresence of VCAsysPC

Capture card	Maximum cahnnels
NCP100	1
NCP1200, NCP2000, NCP3000, NCP3100, NCP3200	4
The other capture card	The number of channel which a board supports

Supported Video Formats and Resolution

	NTSC	PAL
Video Format	VCA5_VIDEO_FORMAT_NTSC_M	VCA 5_VIDEO_FORMAT_PAL_B
Image Size	Minimum width 176 and height 120/Maximum width 1920 and height 1080	
FPS	Over 15FPS	Over 12FPS
Color format	VCA5_COLOR_FORMAT_RGB24 VCA5_COLOR_FORMAT_RGB16 VCA5_COLOR_FORMAT_YUY2 VCA5_COLOR_FORMAT_UYVY VCA5_COLOR_FORMAT_Y8 VCA5_COLOR_FORMAT_YV12	



The most efficient input format is CIF/YUY2. Other formats can be used as input.

Maximum number of supported VCA Engines, Zones, Rules, and Counters

Type	Maximum number
Engine Count	128
License Count	64
Zones	40
Rules	600
Counters	20
SubCounters	20
Classification groups	50
Counting line	5



The raw video image from the capture card should not be changed before input to the VCA engine. If it is changed, the VCA engine may stop working after approximately 5~10 seconds.

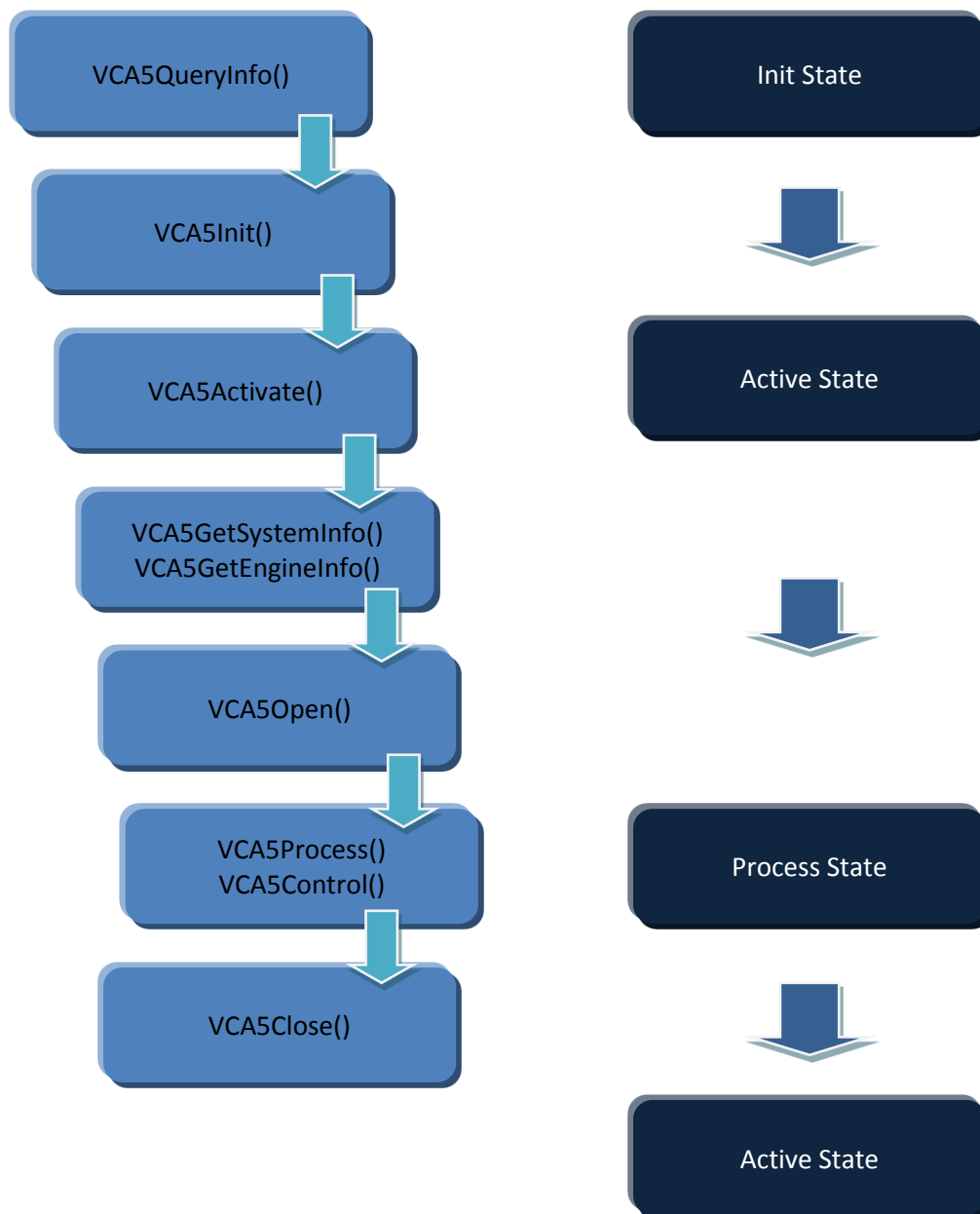
2. Using VCA5 API

2.1. Process state

The API can be in one of three states. The functions that can be called and the operations available vary depending on the API state.

State	Description
INIT	Before the API has been activated, it is possible to query information applicable and necessary for activation.
ACTIVE	After the API has been activated, but before it has been opened, it is possible to retrieve information about the number of supported VCA engines and their features.
PROCESS	Once the API has been successfully activated and opened, it's possible to process video frames.

The following diagram illustrates the overall API process flow and the functions available at each stage.



3. VCA5 API Reference

3.1. Functions

VCA5Init

Initializes the library with the API version.

```
BOOL VCA5Init(  
    unsigned long    ulVersion  
)
```

Parameters

ulVersion

[in] The current version of the VCA5 library. It is defined as VCA5_VERSION in VCA5CoreLib.h.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails or there are no more errors on the error stack, the return value is zero.

Remarks

This function is supported by V1.2 of the SDK only – it's not possible to call it on previous versions.

It's necessary to call this function to activate some specific features available in SDKs V1.2 and above.

Process State

Init

Example Code

```
BOOL rs;  
rs=VCA5Init(VCA5_VERSION);  
if(!rs){  
    VCA5_ERROR_CODE_ITEM errCode;  
    VCA5GetLasError(&errCode);  
    TRACE("Error-%s\n", errCode.AuxMsg);  
}
```

See Also

UDA5 SDK Manual

VCA5GetLastErrorCode

Retrieves the VCA5 API last error code and additional information. The API error codes are maintained in a stack structure. This function provides not only the last error code but also the internal error code information.

```
BOOL VCA5GetLastErrorCode(  
    VCA5_ERROR_CODE_ITEM *pEcode  
)
```

Parameters

pEcode

[out] A pointer to a variable of VCA5_ERROR_CODE_ITEM type to retrieve the error information.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails or there are no more errors on the error stack, the return value is zero.

Remarks

When a VCA5 API function call fails, call VCA5GetLastErrorCode to obtain information related to the error. The VCA5_ERROR_CODE_ITEM structure includes the error code, error time, error display and the session in which the error occurred. This function, unlike other UDA5 APIs, does not require calling VCA5GetLastErrorCode if the return value is FALSE. It simply indicates that there are no more errors.

Unlike GetLastError in the Win32 API, this function continually saves error information in a stack like structure. Therefore, there is no immediate and compulsory need to call this function as soon as an error has occurred.

Process State

Init, Activate, Process

Example Code

```
BOOL rs;  
rs=VCA5Activate(1, pLicenseInfo);  
if(!rs){  
    VCA5_ERROR_CODE_ITEM errCode;  
    VCA5GetLasError(&errCode);  
    TRACE("Error-%s\n", errCode.AuxMsg);  
}
```

See Also

UDA5 SDK Manual

VCA5QueryInfo

Retrieves information about the VCA engine(s) before VCA has been activated.

```
BOOL VCA5QueryInfo(
    unsigned long    uQueryInfoCmd,
    unsigned long    uln,
    void            *pOut
);
```

Parameters

uQueryInfoCmd

[in] Refers to the command to receive special features or attributes that the VCA engine supports.

uln

[in] Refers to the parameters(s) used for each command.

pOut

[out] Query result.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails or there are no more errors on the error stack, the return value is zero.

Remarks

This function is used to retrieve the VCA features or related information before the VCA engine(s) have been activated. In particular, the function is used to retrieve the information necessary to activate the VCA engine(s).

Query command

uQueryInfoCmd	Description
VCA5_QR_GETHWGUID	Retrieves the globally unique hardware identifier (HW GUID) for the capture card(s). The resulting license will be locked to the capture card(s).
VCA5_QR_GETHWGUIDOPEN	Retrieves the globally unique hardware identifier (HW GUID) from the PC on which VCA5 is installed. The resulting license will be locked to the PC.



Note

Use VCA5_QR_GETHWGUID command with **VCA5sysPC**
Use VCA5_QR_GETHWGUIDOPEN with **VCA5sysPCopen**.

Parameters for each query command

uQueryInfoCmd	uln	pOut
VCA5_QR_GETHWGUID VCA5_QR_GETHWGUIDOPEN	The size, in bytes of VCA5_HWGUID_INFO	Pointer to a VCA5_HWGUID_INFO structure

Process State

Init, Activate, Process

Example Code**See Also**

VCA5GetSystemInfo, VCA5GetEngineInfo

VCA5Activate

Activates the VCA engine(s) and enables all VCA5 functions.

```
BOOL VCA5Activate(  
    unsigned long      uLicenseCnt,  
    VCA5_LICENSE_INFO *pVCA5LicenseInfos  
)
```

Parameters

uLicenseCnt

[in] Specifies the number of VCA licenses.

pVCA5LicenseInfos

[in] A pointer to an array of VCA5_LICENSE_INFO structures. Each one corresponds to an individual VCA5 license.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails or there are no more errors on the error stack, the return value is zero.

Remarks

To activate the VCA engine, call **VCA5Activate** with an array of VCA licenses corresponding to the capture boards serial numbers.

Process State

Init

Example Code

```
BOOL InitVCAApi(char* szDllName, char *szDrvDllPath, char *szLicense, char *szUSN)  
{  
    ...  
    VCA5_LICENSE_INFO licenseInfo;  
    licenseInfo.szUSN      = szUSN;  
    licenseInfo.szLicense  = szLicense;  
    licenseInfo.szDrvDllPath = szDrvDllPath;  
    (g_VCAApi->VCA5Activate(1, (VCA5_LICENSE_INFO*)&licenseInfo));  
    ...  
}
```

See Also

VCA5_LICENSE_INFO

VCA5GetSystemInfo

Retrieves the VCA5 system information.

```
BOOL VCA5GetSystemInfo(  
    VCA5_SYSTEM_INFO* pVCA5SystemInfo  
)
```

Parameters

pVCA5SystemInfo

[out] Pointer to a VCA5_SYSTEM_INFO structure to receive the VCA5 system information.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call **VCA5GetLastErrorCode**.

Remarks

VCA5GetSystemInfo function retrieves information about VCA5 system. This includes the number of available VCA engines and API version information. The API must be activated before calling **VCA5GetSystemInfo**.

Process State

Activate, Process

Example Code

```
VCA5Activate(...)  
...  
BOOL rs;  
VCA5_SYSTEM_INFO SysInfo  
rs=VCA5GetSystemInfo(&SysInfo);  
if(!rs){  
    VCA5_ERROR_CODE_ITEM errCode;  
    VCA5GetLasError(&errCode);  
    TRACE("Error-%s\n", errCode.AuxMsg);  
}
```

See Also

VCA5_SYSTEM_INFO, VCA5_ENGINE_INFO

VCA5GetEngineInfo

(DEPRECATED SINCE SDK V1.2)

Retrieves information about a particular VCA engine.

```
BOOL VCA5GetEngineInfo(
    unsigned long      ulEngId,
    VCA5_ENGINE_INFO  *pEngineInfo
)
```

Parameters

dwEngId

[in] Specifies the VCA5 engine Id retrieved by VCA5_SYSTEM_INFO.

pEngineInfo

[out] Pointer to a VCA5_ENGINE_INFO structure to receive the VCA5 engine information.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call **VCA5GetLastErrorCode**.

Remarks

The VCA engine ID is assigned by the VCA5Activate function. The number ordering of the engine ID is the same as VCA5_LICENSE_INFO for VCA5Activate. The engine ID starts from 0.

This function is obsolete from SDK V1.2. If two or more licenses are assigned, it returns the information on the first license.

Process State

Activate, Process

Example Code

```
BOOL rs;
VCA5_SYSTEM_INFO SysInfo
rs=VCA5GetSystemInfo(&SysInfo);
if(!rs){
    VCA5_ERROR_CODE_ITEM errCode;
    VCA5GetLasError(&errCode);
    TRACE("Error-%s\n", errCode.AuxMsg);
}

VCA5_ENGINE_INFO  EngineInfo;
for(i=0;i< EngineInfo. uNumOfEngine;i++) {
    g_VCAApi->VCA5GetEngineInfo(i,&EngineInfo);
    DbgMsg("Eng[%d] LicenseNum[%d] Featuretype[%d] Function [0x%08X] \n",
        i, EngineInfo. ulLicenseNum,, EngineInfo. ulFeatureType, EngineInfo.uFunction);
}
```

See Also

VCA5_SYSTEM_INFO, VCA5_ENGINE_INFO

VCA5Open

Initializes an instance of a VCA5 engine with the specified parameters.

```
BOOL VCA5Open(  
    unsigned long          uEngId,  
    VCA5_ENGINE_PARAMS *pEngineParam  
)
```

Parameters

uEngId

[in] Specifies the VCA5 engine ID retrieved by VCA5_SYSTEM_INFO.

pEngineParams

[in] A pointer to a VCA5_ENGINE_PARAMS structure to set VCA engine parameters. These include the video standard, size, color format, resolution, frame rate, etc.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call **VCA5GetLastErrorCode**.

Remarks

For detailed information about the available parameter, refer to Section 1.3.

Process State

Activate, Process

Example Code

```
VCA5_ENGINE_PARAMS  EngineParam;  
EngineParam.ulColorFormat  = VCA5_COLOR_FORMAT_YV12;  
EngineParam.ulVideoFormat  = VCA5_VIDEO_FORMAT_PAL_B;  
EngineParam.ulImageSize    = VCA5_IMAGESIZE_720X576;  
EngineParam.ulFrameRate= 25;  
for(i=0;i<g_nVCAEngine;i++) {  
    g_VCAApi->VCA5Open(i,&EngineParam);  
}
```

See Also

VCA5_ENGINE_PARAMS, VCA5Close

VCA5Close

Closes an instance of an open VCA5 engine.

```
BOOL VCA5Close(  
    unsigned long    ulEngId,  
)
```

Parameters

ulEngId

[in] Specifies the VCA5 engine Id retrieved by VCA5_SYSTEM_INFO.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call **VCA5GetLastErrorCode**.

Remarks

Call this function to close a VCA engine that was previously opened.

Process State

Process

Example Code

See Also

VCA5Open

VCA5Process

Passes a frame of video to an instance of a VCA engine for processing. The exact nature of the analytics functions performed on the video frame will depend on the type of license installed for that engine.

```

BOOL VCA5Process(
    unsigned long      ulEngId,
    unsigned char      *pImage,
    VCA5_TIMESTAMP     *pTimestamp,
    unsigned long      *uiLengthResult,
    unsigned char      *pResult
)

```

Parameters

ulEngId

[in] Specifies the VCA5 engine Id retrieved by VCA5_SYSTEM_INFO.

pImage

[in] Pointer to an image to process. The image format must be compatible with that previously specified in **VCA5Open**.

pTimestamp

[in] Pointer to a VCA5_TIMESTAMP structure corresponding to the localtime timestamp of the input image. VCA5_TIMESTAMP is the same as the FILETIME structure in the Win32 API.

uiLengthResult

[in, out] Pointer to the length of the input buffer, *pResult*. If the function call is successful, this variable is filled with the length of valid metadata when the function returns. The exact format of the metadata will depend on the functions activated for that VCA engine.

pResult

[in, out] Pointer to a buffer which holds the result of the VCA frame analysis. The exact format of the output depends on the features activated for the VCA engine. For VCAsys, the buffer is an XML document. For VCAstba, the buffer is a stabilized video frame.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call **VCA5GetLastErrorCode**.

Remarks

The timestamp is required internally by the VCA engine. The timestamp may be passed from that retrieved by the Cap5 API but it must first be converted to localtime.

VCA5Process uses Intel SSE technology so pImage parameter must be aligned to 16bit

Process State

Process

Example Code

```
FILETIME timestamp;
```

```
BYTE    pResult[VCA5_MAX_OUTPUT_BUF_SIZE];
DWORD   nResult = VCA5_MAX_OUTPUT_BUF_SIZE;

GetSystemTimeAsFileTime(&timestamp);
FileTimeToLocalFileTime(&timestamp, &timestamp);
g_VCAApi->VCA5Process(eng,pImage,(VCA5_TIMESTAMP*)&timestamp,
                    &nResult, pResult));
```

See Also

VCA5Open, VCA5Close, VCA5Control
VCAsys XML metadata format.pdf.

VCA5Control

Controls an instance of a VCA engine to configure additional properties, for example detection rules, stabilization parameters, etc.

```

BOOL VCA5Control(
    unsigned long    ulEngId,
    unsigned long    ulCmd,
    VCA_ULONGPTR ulParam0,
    VCA_ULONGPTR ulParam1,
    VCA_ULONGPTR ulParam2,
    VCA_ULONGPTR ulParam3
)

```

Parameters

ulEngId

[in] Specifies the VCA5 engine ID retrieved by VCA5_SYSTEM_INFO.

ulCmd

[in] Specifies a command to configure a VCA engine. The parameters are dependent on the value of *ulCmd*.

ulParam0

[in] Additional parameter, varies according to *ulCmd*.

ulParam1

[in] Additional parameter, varies according to *ulCmd*.

ulParam2

[in] Additional parameter, varies according to *ulCmd*.

ulParam3

[in] Additional parameter, varies according to *ulCmd*.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call **VCA5GetLastErrorCode**.

Remarks

The values of *ulParam0* to *ulParam3* are dependent on the value of *ulCmd*. Unused parameters are ignored by the VCA API.

VCA command

Command	Meaning
VCA5_CMD_SETZONE	Sets a zone.
VCA5_CMD_GETZONE	Retrieves a zone (e.g. when an XML config has been loaded).
VCA5_CMD_CLEARZONE	Clears a zone.
VCA5_CMD_SETRULE	Sets a rule.
VCA5_CMD_GETRULE	Retrieves a rule (e.g. when an XML config has been

	loaded).
VCA5_CMD_CLEARRULE	Clears a rule.
VCA5_CMD_SETSYNCTIME	Resynchronizes the VCA engine time. Since the VCA engine is isolated (it does not call outside its own DLL), it does not know the real time when generating events.
VCA5_CMD_SETCOUNTER	Sets a counter.
VCA5_CMD_GETCOUNTER	Retrieves a counter (e.g. when an XML config has been loaded).
VCA5_CMD_CLEARCOUNTER	Clears a counter.
VCA5_CMD_SETMETAFMT	Sets the metadata format.
VCA5_CMD_GETMETAFMT	Retrieves the metadata format.
VCA5_CMD_SETCALIBPARAMS	Sets camera calibration parameters.
VCA5_CMD_GETCALIBPARAMS	Gets the camera calibration parameters.
VCA5_CMD_SETOBJECT	Sets an object classification definition.
VCA5_CMD_CLEAROBJECT	Clears an object classification definition.
VCA5_CMD_GETOBJECT	Retrieves an object classification definition.
VCA5_CMD_SETRETRIGTIME	Sets the alarm retrigger time, in seconds. The default value is 0. The alarm retrigger time determines the time between successive alarm events for the same object/rule combination. For example, if an object triggers a zone with a presence rule configured, an event will be generated. If the object then leaves that zone and re-enters it quickly, another event will only be generated if the time since the first event is greater than the alarm retrigger time.
VCA5_CMD_GETRETRIGTIME	Retrieves the alarm retrigger time, in seconds.
VCA5_CMD_SETSTABENABLE	Switches camera shake cancellation on/off. The default setting for camera shake cancellation is off. If the camera is likely to suffer false alarms due to camera sway, use this command to enable camera shake cancellation.
VCA5_CMD_GETSTABENABLE	Retrieves the current setting of camera shake cancellation.
VCA5_CMD_SETTAMPERPARAMS	Sets the tamper detection parameters.
VCA5_CMD_GETTAMPERPARAMS	Retrieves the tamper detection parameters.
VCA5_CMD_SAVECFGXML	Save the current VCA5 configuration to XML file or buffer. This can be reloaded at a later date by VCA5_CMD_LOADCFGXML.
VCA5_CMD_LOADCFGXML	Load a VCA5 configuration from a file or buffer.
VCA5_CMD_SETTRACKERPARAMS	Set the tracking parameters.
VCA5_CMD_SETSCENECHANGEPARAMS	Set the scene change parameters.

Parameters for each VCA command

Command	Parameter
VCA5_CMD_SETZONE	ulParam0: [in] Pointer to a VCA5_ZONE structure. ulParam1, ulParam2, ulParam3: not used.

VCA5_CMD_GETZONE	ulParam0: [in] Specifies the zone ID to retrieve. ulParam1: [out] Pointer to a VCA5_ZONE structure. ulParam2, ulParam3: not used.
VCA5_CMD_CLEARZONE	ulParam0: [in] Specifies the zone ID to clear. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_SETRULE	ulParam0: [in] Pointer to a VCA5_RULE structure. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_GETRULE	ulParam0: [in] Specifies the rule ID to retrieve. ulParam1: [out] Pointer to a VCA5_RULE structure. ulParam2, ulParam3: not used.
VCA5_CMD_CLEARRULE	ulParam0: [in] Specifies the rule ID to clear. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_SETSYNCTIME	ulParam0: [in] Pointer to a VCA5_TIME_STRUCT structure. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_SETCOUNTER	ulParam0: [in] Pointer to a VCA5_COUNTER structure. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_GETCOUNTER	ulParam0: [in] Specifies the counter ID to retrieve. ulParam1: [out] Pointer to a VCA5_COUNTER structure. ulParam2, ulParam3: not used.
VCA5_CMD_CLEARCOUNTER	ulParam0: [in] Specifies the counter ID to clear. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_SETMETAFMT	ulParam0: [in] Specifies the metadata format. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_GETMETAFMT	ulParam0: [out] Pointer to an integer to receive the value of metadata format. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_SETCALIBPARAMS	ulParam0: [in] Pointer to a VCA5_CALIB_INFO structure. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_GETCALIBPARAMS	ulParam0: [in] it must be zero. ulParam1: [out] Pointer to a VCA5_CALIB_INFO structure. ulParam2, ulParam3: not used.
VCA5_CMD_SETOBJECT	ulParam0: [in] Pointer to a VCA5_CLSOBJECT structure. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_GETOBJECT	ulParam0: [in] Specifies the classification object ID to retrieve. ulParam1: [out] Pointer to a VCA5_CLSOBJECT structure. ulParam2, ulParam3: not used.
VCA5_CMD_CLEAROBJECT	ulParam0: [in] Specifies the object ID to clear. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_SETRETRIGTIME	ulParam0: [in] Specifies the alarm retrigger time, in seconds. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_GETRETRIGTIME	ulParam0: [out] Pointer to an integer to receive the value of retrigger time, in seconds. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_SETSTABENABLE	ulParam0: [in] Specifies whether to enable camera shake cancellation. 1 = ON, 0 = OFF. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_GETSTABENABLE	ulParam0: [out] Pointer to an integer that specifies

	whether camera shake is enabled. 1 = ON, 0 = OFF. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_SETTAMPERPARAMS	ulParam0: [in] Pointer to a VCA5_TAMPER_INFO structure. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_GETTAMPERPARAMS	ulParam0: [out] Pointer to a VCA5_TAMPER_INFO structure. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_SAVECFGXML	ulParam0: [in] Pointer to a VCA5_XMLCFG_PARAMS structure. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_LOADCFGXML	ulParam0: [in] Pointer to a VCA5_XMLCFG_PARAMS structure. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_SETTRACKERPARAMS	ulParam0: [in] Pointer to a VCA5_TRACKERPARAMS structure. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_SETSCENECHANGE_PARAMS	ulParam0: [in] Pointer to a VCA5_SCENECHANGE_INFO structure. ulParam1, ulParam2, ulParam3: not used.
VCA5_CMD_GETSCENECHANGE_PARAMS	ulParam0: [out] Pointer to a VCA5_SCENECHANGE_INFO structure. ulParam1, ulParam2, ulParam3: not used.

For the commands VCA5_CMD_CLEARZONE, VCA5_CMD_CLEARRULE, and VCA5_CMD_CLEARCOUNTER ulParam0 may be set to VCA5_ID_ALL. This removes all zone, rule, and counter settings.

Process State

Process

Example Code

See Also

VCA5Open, VCA5Close, VCA5Control

3.2. Structures

VCA5_HWGUID_INFO

Contains hardware GUID information correspond to a PCVCA capture board

```
char*      szUSN;  
char*      szGUID;  
char*      szDrvDllPath;  
unsigned long reserved[VCA5_MAX_RESERVED_HWGUID_INFO];  
} VCA5_HWGUID_INFO;
```

Members

szUSN

The unique serial number of the board.

szGUID

The GUID that corresponds to the USN of the capture board, or the PC, depending on whether VCA5_QR_GETHWGUID or VCA5_QR_GETHWGUIDOPEN was passed to the corresponding call to VCA5Query.

szDrvDllPath

The dll path of the driver. It is used to communicate with the capture board using the Cap5 interface.

reserved

Reserved for future use.

Remarks

Used to activate the VCA5 engines and query the hardware GUID. If multiple cards are installed, VCA5_HWGUID_INFO contains the information of the first detected capture board.

In case of **VCA5sysPCopen**, *szUSN* and *szDrvDllPath* members are not required and should be set to NULL.

VCA5_LICENSE_INFO

Contains license information for activating the VCA system

```
typedef struct _VCA5_LICENSE_INFO {
    char*          szUSN;
    char*          szLicense;
    char*          szDrvDllPath;
    unsigned char  nLicenseID;          //Out
    unsigned long  ulNumOfEngine;      //Out
    unsigned long  ulFunction;         //Out
    char          szLicenseDesc[VCA5_MAX_STR_LEN]; //Out
    unsigned long  reserved[VCA5_MAX_RESERVED_LICENSE_INFO];
} VCA5_LICENSE_INFO;
```

Members

szUSN

The unique serial number of the board.

szLicense [In]

The activation code that corresponds to the USN of the capture board, or to the PC in the case of VCA5sysPCopen

szDrvDllPath [In]

The dll path of the driver. It is used to communicate with the capture board using the Cap5 interface.

nLicenseID [Out]

The assigned License ID by VCA5Lib. It is used VCA5Open in VCA5_ENGINE_PARAMS structure to assign using license.

ulNumOfEngine [Out]

The number of available engines that license supports

ulFunction [Out]

The available functions that VCA engine supports.

Funtions	Meaning
VCA5_FEATURE_PRESENCE	Supports Presence filter
VCA5_FEATURE_ENTER	Supports Enter filter
VCA5_FEATURE_EXIT	Supports Exit filter
VCA5_FEATURE_APPEAR	Supports Appear filter
VCA5_FEATURE_DISAPPEAR	Supports Disappear filter
VCA5_FEATURE_STOPPED	Supports Stop filter
VCA5_FEATURE_DWELL	Supports Dwell filter
VCA5_FEATURE_DIRECTION	Supports Direction filter
VCA5_FEATURE_SPEED	Supports Speed filter
VCA5_FEATURE_ABOBJ	Supports Abandon Object filter
VCA5_FEATURE_TAILGATING	Supports Tailgating filter
VCA5_FEATURE_LINECOUNTER	Supports Line Counter filter
VCA5_FEATURE_SMOKE	Supports Smoke filter
VCA5_FEATURE_FIRE	Supports Fire filter
VCA5_FEATURE_COUNTING	Supports Counting function

VCA5_FEATURE_CALIBRATION	Supports Camera camera calibration.
VCA5_FEATURE_METADATA	Supports metadata output
VCA5_FEATURE_PEOPLETRACKING	Supports people tracker engine

szDrvDllPath

The dll path of the driver. It is used to communicate with the capture board using the Cap5 interface.

reserved

Reserved for future use.

Remarks

Used to activate the VCA5 engines and query the hardware GUID.

In the case of VCAPresence, a license code of NULL can be passed. The license for VCAPresence is burnt into the capture board.

In case of **VCA5sysPCopen**, set *szUSN* and *szDrvDllPath* members to NULL.

From SDK V1.2, in order to get the activated license information, VCA5_LICENSE_INFO is used instead of VCA5_ENGINE_INFO because of the multi-license support.

VCA5_SYSTEM_INFO

Used to retrieve information about the VCA system, after it has been activated.

```
typedef struct _VCA5_SYSTEM_INFO {  
    unsigned long    ulNumOfLicense;  
    unsigned long    ulNumOfEngine[VCA5_MAX_NUM_LICENSE];  
    unsigned long    ulDllVersion;  
    unsigned long    reserved[VCA5_MAX_RESERVED_SYSTEM_INFO];  
} VCA5_SYSTEM_INFO;
```

Members

ulNumOfLicense

The total number of license to support VCA system after activation.

ulNumOfEngine

The number of available engines that each license supports.

ulDllVersion

The version of VCA5Lib.dll

reserved

Reserved for future use.

Remarks

VCA5_ENGINE_INFO

(DEPRECATED SINCE SDK V1.2)

Used to retrieve information about an instance of a VCA engine.

```
typedef struct _VCA5_ENGINE_INFO {
    unsigned long    ulLicenseNum;
    unsigned long    ulFeatureType;
    unsigned long    ulFunction;
    char             szLicenseDesc[VCA5_MAX_LICENSE_DESC_LEN];
    unsigned long    reserved[VCA5_MAX_RESERVED_ENGINE_INFO];
} VCA5_ENGINE_INFO;
```

Members

ulLicenseNum

The index of the license that corresponds to the VCA engine ID.

ulFeatureType

The available VCA feature type, only VCA5_FEATURE_DEF_VCASYS is supported now.

Funtions	Meaning
VCA5_FEATURE_PRESENCE	Support Presence filter
VCA5_FEATURE_ENTER	Support Enter filter
VCA5_FEATURE_EXIT	Support Exit filter
VCA5_FEATURE_APPEAR	Support Appear filter
VCA5_FEATURE_DISAPPEAR	Support Disappear filter
VCA5_FEATURE_STOPPED	Support Stop filter
VCA5_FEATURE_DWELL	Support Dwell filter
VCA5_FEATURE_DIRECTION	Support Direction filter
VCA5_FEATURE_SPEED	Support Speed filter
VCA5_FEATURE_ABOBJ	Support Abandoned/Removed Object filter
VCA5_FEATURE_TAILGATING	Support Tailgating filter
VCA5_FEATURE_LINECOUNTER	Support Line Counter filter
VCA5_FEATURE_SMOKE	Support Smoke Object filter
VCA5_FEATURE_FIRE	Support Fire Object filter
VCA5_FEATURE_COUNTING	Support Counting function
VCA5_FEATURE_CALIBRATION	Support Camera calibration.
VCA5_FEATURE_METADATA	Support metadata output

uFunction

The available functions that VCA engine supports.

szLicenseDesc

The license description.

reserved

Reserved for future use.

Remarks

This structure is obsolete from SDK V1.2. If two or more licenses are assigned, it returns the information on the first license.

VCA5_ENGINE_PARAMS

Used when creating a new instance of a VCA engine to set the engine parameters.

```
typedef struct _VCA5_ENGINE_PARAMS {  
    unsigned long    ulVideoFormat;  
    unsigned long    ulColorFormat;  
    unsigned long    ulImageSize;  
    unsigned long    ulFrameRate100;  
    unsigned long    ulLicenseCnt;  
    unsigned char    ucLicenseId[VCA5_MAX_NUM_LICENSE];  
    VCA5_RECT        imageROI;  
    unsigned long    reserved[VCA5_MAX_RESERVED_ENGINE_PARAM-5];  
} VCA5_ENGINE_PARAMS;
```

Members

ulVideoFormat

The video standard, e.g. PAL or NTSC.

ulColorFormat

The video color format.100

ullImageSize

The size of the image.

ulFrameRate100

The framerate multiplied by 100 (necessary in order to represent fractional frame rates e.g 12.5fps).

This parameter does not used, because timestamp used for checking frame interval.

ulLicenseCnt

The number of using license.

ucLicenseId

The licenseid array by activated.

imageROI

Select ROI to work VCA (experimental).

reserved

Reserved for future use.

Remarks

For detailed information about the available parameter, refer to section “1.5. Specification”

The license should be assigned when opening a engine so as to support the multi-license a engine.

ulFrameRate100 does not used anymore and replaced by timestamp in VCA5Process.

VCA5_ZONE

Used to configure a zone with VCA5_CMD_SET_ZONE and VCA5_CMD_GETZONE.

```
typedef struct _VCA5_ZONE {
    unsigned short    usZoneId;
    unsigned short    usZoneType;
    unsigned short    usZoneStyle;
    unsigned long     ulTotalPoints;
    VCA5_POINT        pPoints[VCA5_MAX_NUM_ZONE_POINTS];
    char              szZoneName[VCA5_MAX_STR_LEN];
}VCA5_ZONE;
```

Members

usZoneId

A unique zone ID. Must not exceed VCA5_MAX_NUM_ZONES (40).

usZoneType

Specifies the type of the zone.

Zone Type	Description
VCA5_ZONE_TYPE_DETECTION	Detection zone which is configured for use by detection rules.
VCA5_ZONE_TYPE_NONDETECTION	Non-detection zone. Nothing will be detected in a non-detection zone.

usZoneStyle

Specifies the style of the zone.

Zone Style	Description
VCA5_ZONE_STYLE_TRIPWIRE	The zone is configured as a detection line.
VCA5_ZONE_STYLE_POLYGON	The zone is configured as a polygonal detection area.
VCA5_ZONE_STYLE_NOTDEFINED	Zone is not defined.

ulTotalPoints

The number of polygon vertices.

pPoints

The list of polygon vertices, normalized across 16 bits.

szZoneName

Zone name.

Remarks

The list of polygon vertices is a list of VCA5_POINTS. The x and y co-ordinates in each point are normalized across 16 bits in order to avoid specialized handling for different video sizes. The example code below illustrates how to denormalize a normalized point.

```
#define NORMALISATOR 65535

// maxValue : real size of image
#define PERCENTTOPIXEL( pixelValue, normalizeValue, maxValue ) \
    pixelValue = (( maxValue - 1 ) * normalizeValue) / NORMALISATOR;
```


VCA5_RULE

Used to set the properties of a detection rule when used with the VCA5_CMD_SETRULE and VCA5_CMD_GETRULE command.

```
typedef struct _VCA5_RULE{
    unsigned long        ulRuleId;
    unsigned short       usRuleType;
    unsigned short       usTotalTrkObjs;
    unsigned char        ucTrkObjs[VCA5_MAX_NUM_CLSOBJECTS+2];
    unsigned short       usZoneId;

    union RULE_DATA{
        VCA5_RULE_TIMER        tTimer;
        VCA5_RULE_DIRECTION    tDirection;
        VCA5_RULE_SPEED        tSpeed;
        VCA5_RULE_AREA         tArea;
        VCA5_RULE_LINECOUNTER  tLineCounter;
        VCA5_RULE_RESERVED     tReserved;
    }tRuleDataEx;

    char                szRuleName[VCA5_MAX_STR_LEN];
    unsigned long       reserved[VCA5_MAX_RESERVED_ENGINE_PARAM];
} VCA5_RULE;
```

Members

ulRuleId

A unique rule ID. Must not exceed VCA5_MAX_NUM_RULES (600).

usRuleType

Specifies the type of rule to set. For detailed information, refer to “3.3. Rule type”

usTotalTrkObjs

The total number of tracking object.

ucTrkObjs

The specific object classifier id to include or to exclude for tracking

7	6	5	4	3	2	1	0
0 : include		Object classifier ID					
1 : exclude							

usZoneId

The id of the zone upon which the detection rule will operate.

tRuleDataEx

Extended rule configuration parameters such as time and direction.

tTimer

Extended rule data for those rules that require some notion of timing. E.g. Stopped or Dwell filters.

tDirection

Extended rule data for those rules that require a directional value, e.g. Direction filter.

tSpeed

Extended rule data for those rules that require a lower and upper bound speed. The measurement units depend on the calibration configuration (mph or km/h). E.g. Speed filter

tArea

Extend rule data for those rules that require an area value. The measurement units depend on the calibration configuration (m² or square feet).

tLineCounter

Extended rule data for those rules that require a LineCounter value. You can choose both width calibration and shadow filter.

szRuleName

Rule name.

reserved

Reserved for future use.

Remarks

Rule type	Description	Extend parameter
VCA5_RULE_TYPE_PRESENCE	Detects the presence of an object in a detection zone.	
VCA5_RULE_TYPE_ENTER	Detects objects entering a detection zone.	
VCA5_RULE_TYPE_EXIT	Detects objects exiting a detection zone.	
VCA5_RULE_TYPE_APPEAR	Detects objects appearing in a detection zone.	
VCA5_RULE_TYPE_DISAPPEAR	Detects objects disappearing in a detection zone.	
VCA5_RULE_TYPE_STOP	Detects objects that have stopped in a detection zone for the specified amount of time.	VCA5_RULE_TIMER Threshold time in seconds.
VCA5_RULE_TYPE_DWELL	Detects objects that have dwelled in a detection zone for the specified amount of time.	VCA5_RULE_TIMER Threshold time in seconds.
VCA5_RULE_TYPE_DIRECTION	Detects objects travelling in a zone in the specified direction.	VCA5_RULE_DIRECTION Specify the start and finish angles in tenths of a degree (0 to 3600).
VCA5_RULE_TYPE_SPEED	Detects object travelling within the bounds of the configured speeds	VCA5_RULE_SPEED Threshold speed in mph or km/h (depends on the calibration configuration).
VCA5_RULE_TYPE_TAILGATING	Detects an object crossing a line or zone within a certain time after an object has already crossed the line or zone.	VCA5_RULE_TIMER Threshold time between two objects crossing line or zone in seconds.

VCA5_RULE_TYPE_LINECOUNT ER_A	Raises an event when an object crosses the line in the A direction.	VCA5_RULE_LINECOUNT ER Configuration of width calibration and shadow filter
VCA5_RULE_TYPE_LINECOUNT ER_B	Raises an event when an object crosses the line in the B direction.	VCA5_RULE_LINECOUNT ER Configuration of width calibration and shadow filter
VCA5_RULE_TYPE_ABOBJ, VCA5_RULE_TYPE_RMOBJ	Raises an event when an object is abandoned or removed in detection zone.	VCA5_RULE_TIMER Threshold time in seconds.
VCA5_RULE_TYPE_SMOKE	Raises an event when smoke is detected	VCA5_RULE_TIMER Threshold time in seconds.
VCA5_RULE_TYPE_FIRE	Raises an event when fire is detected	VCA5_RULE_TIMER Threshold time in seconds.
VCA5_RULE_TYPE_COLSIG	Raises an event when color exceeds threshold	VCA5_RULE_COLSIG Specific color threshold percent in object

VCA5_RULE_LINECOUNTER

Used to set the properties of the extra parameter for Linecounter.

```
typedef struct _VCA5_RULE_LINECOUNTER{  
    unsigned long        ulCalibrationWidth;  
    unsigned long        ulShadowFilterEnabled;  
}VCA5_RULE_LINECOUNTER;
```

Members

ulCalibrationWidth

Set the calibration width of a single object in normalized value across 65535 as the width of the image. If the value is set to zero, the line is not calibrated and may count connected objects as a single object.

ulShadowFilterEnabled

Set to use the shadow filter on the counting line. When set, the counting line filters out shadows cast by humans and other objects (e.g. rotating doors).



Only enable the shadow filter if there is a specific problem with shadows where the counting line is to be used. It is recommended that the shadow filter only be enabled when shadows are present because the algorithm can mistake certain parts of an object for shadows and this may lead to worse counting results.

VCA5_RULE_TIMER

Used to set the properties of the extra parameter for Stop/Dwell, Tailgating, Abandoned /Removed, Tailgating and Smoke/Fire filter.

```
typedef struct _VCA5_RULE_TIMER{
    unsigned long    ulTimeThreshold;
    unsigned short   usZoneld;           // 16-bit
    unsigned short   usFlags;           // 16-bit, extra flags for rules
}VCA5_RULE_TIMER;
```

Members

ulTimeThreshold

Threshold time.

usZoneld

Not used for Stop, Dwell and Tailgating filters.

usFlags

Rule-specific flags required depending on the rule.

Remarks

If the rule is the tailgating filter, usFlags specifies the engine type as the trigger source.

VCA5_RULE_TAILGATING_BYTRACKER : trigger tailgating rule by tracker

VCA5_RULE_TAILGATING_BYLINECOUNTER: trigger tailgating rule by line counter

VCA5_RULE_COLSIG

Used to set the properties of the extra parameter for color filter.

```
typedef struct _VCA5_RULE_COLSIG{
    unsigned short   usColBin;
    unsigned short   usThreshold;
}VCA5_RULE_COLSIG;
```

Members

ulColBin

Index VCA5_PALLET_COLOR_E in header

black(0), brown(1), grey(2), blue(3), green(4), cyan(5), red(6), magenta(7), yellow(8), white(9).

usThreshohold

Percent color threshold

Remarks

Colors cannot be detected in case the size of an object is too small. Additionally, the color information is not sourced from the bounding box, but from the blobs of an object.

VCA5_SUBCOUNTER

Used when setting the parameters of a sub counter with the VCA5_CMD_SET_SUBCOUNTER command. A sub counter connects a rule to a counter. Every time the rule generates an event, the sub counter notifies a counter according to its configuration (increment, decrement, instantaneous value, etc).

```
typedef struct _VCA5_SUBCOUNTER{
    unsigned short    usSubCounterType;
    unsigned short    usTrigId;
}VCA5_SUBCOUNTER;
```

Members

usSubCounterType

Specifies the type of the sub counter.

usTrigId

Specifies the rule ID that, when triggered, will cause the counter to be modified according to the type of the sub counter.

Remarks

usSubcounterType	Description
VCA5_COUNTER_INCREMENT	The count is incremented every time an event is raised by the rule being triggered.
VCA5_COUNTER_DECREMENT	The count is decremented every time an event is raised by the rule being triggered.
VCA5_COUNTER_INSTANT	The count is set to the instantaneous value of the number of objects that have triggered the rule. E.g. if a presence rule is configured on a particular zone, there may be 10 objects that trigger the rule.

VCA5_COUNTER

Used to set the properties of a counter with the VCA5_CMD_SET_COUNTER or VCA5_CMD_GET_COUNTER command.

```
typedef struct _VCA5_COUNTER{
    unsigned short    usCounterId;
    unsigned short    usNumSubCounters;
    VCA5_SUBCOUNTER  pSubCounters[VCA5_MAX_NUM_SUBCOUNTERS];
    char              szCounterName[VCA5_MAX_STR_LEN];
}VCA5_COUNTER;
```

Members

usCounterId

A unique counter ID. Must not exceed VCA5_MAX_NUM_RULES (600).

usNumSubCounters

The number of sub counters in the pSubCounters array. Must not exceed VCA5_MAX_NUM_SUBCOUNTERS(20).

pSubCounters

An array of VCA5_SUBCOUNTER structures to assign to the counter.

szCounterName

The counter name.

Remarks

A counter is provided inputs by sub counters. The counter displays the overall result of the addition of all of its sub counter inputs. E.g. if a zone is configured with enter and exit rules and a sub counter is tied to each rule with the enter rule having a VCA5_COUNTER_INCREMENT type and the exit rule having a VCA5_COUNTER_DECREMENT_TYPE, these can both be assigned to a counter. The counter would then display the total number of objects that had entered the zone, minus the number of objects that exited, in effect providing an occupancy function.

VCA5_CALIB_INFO

Used to set the engine calibration parameters with the VCA5_CMD_SETCALIBPARAMS or VCA5_CMD_GETCALIBPARAMS command.

```
typedef struct _VCA5_CALIB_INFO{
    float      fHeight;
    float      fTilt;
    float      fRoll;
    float      fFOV;
    int         calibrationStatus;
    unsigned long ulHeightUnits;
    unsigned long ulSpeedUnits;
}VCA5_CALIB_INFO;
```

Members

fHeight

Height of camera installed in meters

fTilt

Camera title angle in meters

fRoll

Camera rolling angle. Not supported yet.

fFOV

Vertical field of view angle of camera

calibrationStatus

Calibration result returned after calibration.

ulHeightUnits

The units to be used for object height measurement, can be one of VCA5_HEIGHT_UNITS_METERS or VCA5_HEIGHT_UNITS_FEET.

ulSpeedUnits

The units to be used for object speed measurement, can be one of VCA5_SPEED_UNITS_KPH or VCA5_SPEED_UNITS_MPH.

Remarks

Calibration status	Description
VCA5_CALIB_STATUS_CALIBRATED_OVERHEAD	Calibration successful in overhead mode.
VCA5_CALIB_STATUS_CALIBRATED_SIDEON	Calibration successful in side-on mode.
VCA5_CALIB_STATUS_CALIBRATED	Calibration successful.
VCA5_CALIB_STATUS_UNCALIBRATED	The initial value when the engine has not been calibrated.
VCA5_CALIB_STATUS_INVALIDSETUP	Calibration failed.
VCA5_CALIB_STATUS_FAILED_TOOFEW_MARKERS	Obsolete.
VCA5_CALIB_STATUS_FAILED_TILT_OUTOFRANGE	Calibration failed because tilt angle parameter is out of range.
VCA5_CALIB_STATUS_FAILED_HEIGHT_OUTOFRANGE	Calibration failed because height parameter is out of range.

VCA5_CALIB_STATUS_FAILED_FOV_OUTOFRANGE	Calibration failed because fov(vertical field of view) parameter is out of range.
---	---

VCA5_CLSOBJECT

Used to configure a class object with VCA5_CMD_SETOBJECT and VCA5_CMD_GETOBJECT.

```
typedef struct _VCA5_CLSOBJECT{
    short          sClsObjectId;
    VCA5_RULE_AREA tAreaSetting;
    VCA5_RULE_SPEED tSpeedSetting;
    char           szClsObjectName[ VCA5_MAX_STR_LEN ];
} VCA5_CLSOBJECT;
```

Members

sClsObjectId

A unique counter ID. Must not exceed VCA5_MAX_NUM_CLSOBJECTS (20).

tAreaSetting

area of area defined by VCA5_RULE_AREA.

tSpeedSetting

object size define by VCA5_RULE_SPEED.

szClsObjectName

Class object name.

Remarks

VCA5sys can perform object classification once the camera has been calibrated. The object classification is based on properties extracted from the object including object area and speed. VCA5sys comes pre-loaded with the most common object classes, and in most cases these will not need to be modified. Objects that do not fit into any class are labeled as "Unclassified".

If the classification definition of an object is overlapped, it is classified by the object class defined ahead.

VCA5_TAMPER_INFO

Used to set the tamper detection parameters with the VCA5_CMD_SETTAMPERPARAMS or VCA5_CMD_GETTAMPERPARAMS command.

```
typedef struct _VCA5_TAMPER_INFO {  
    unsigned long    ulEnabled;  
    unsigned long    ulAlarmTimeout;  
    unsigned long    ulAreaThreshold;  
    unsigned long    ulFiringThreshold;  
    unsigned long    ulLowLightEnabled;  
} VCA5_TAMPER_INFO;
```

Members

ulEnabled

Enable/Disable tamper detection. 0 is 'enable' and 1 is 'disable'.

ulAlarmTimeout

The length of time (in seconds) that the image must be persistently changed before the alarm is triggered.

ulAreaThreshold

Percentage area of the image which must be changed for the tamper alarm to be triggered.

ulFiringThreshold

Not used.

ulLowLightEnabled

Enable/Disable the option to prevent false alarm due to large fast changes to the image lighting such as switching on/off indoor lighting. Enable this option if this is likely to be a problem in the area where the camera is installed. However, this option will reduce sensitivity to genuine alarms so it is not recommended if fast light changes are not likely to be a problem.

Remarks

If false alarms are a problem, the duration and/or area should be increased so that large transient changes such as a close object temporarily obscuring the camera do not cause false alarms.

VCA5_XMLCFG_PARAMS

```
typedef struct _VCA5_XMLCFG_PARAMS {  
    unsigned long    ulMedia;           // Save/load to file or buffer  
    unsigned long    ulCfgFlags;       // Flags say what type of info to save/load  
    char             *pszBufOrFilename; // The buffer or the filename  
    unsigned long    ulBufLen;         // How long is the buffer (in bytes)  
} VCA5_XMLCFG_PARAMS;
```

Members

ulMedia

Determines whether to load/save from/to a memory buffer or a file. To specify a buffer pass VCA5_XMLCFG_BUFFER, and to specify a file pass VCA5_XMLCFG_FILE.

ulCfgFlags

A bitmask of VCA5_XMLCFG_FLAGS that specifies which parts of the configuration to load/save. Specify VCA5_CFGFLAG_ALL to load/save everything.

pszBufOrFilename

A pointer to a buffer or a string containing the filename to save to.

ulBufLen

The length of the buffer, in bytes, passed in pszBufOrFilename. Only valid when ulMedia is set to VCA5_XMLCFG_BUFFER.

VCA5_TRACKERPARAMS

```
typedef struct _VCA5_TRACKERPARAMS{
    unsigned long    ulMinObjAreaPix;
    unsigned long    ulSecsToHoldOn;
    unsigned long    bAbObjEnabled;
    unsigned long    bMovObjEnabled;
    unsigned long    bAdvTraEnabled;
    unsigned long    bCntLineEnabled;
    unsigned long    bSmokeFireEnabled;
    unsigned long    ulDetectionPoint;
    unsigned long    reserved[VCA5_MAX_RESERVED_TRACKERPARAMS];
}VCA5_TRACKERPARAMS;
```

Members

ulMinObjAreaPix

The minimum area (in blob-pixels) of objects that should be considered for tracking.

ulSecsToHoldOn

The number of seconds to hold on to an object once it becomes stationary and burning it into the background (stop tracking it and assume it has become a part of the scene).

bAbObjEnabled

Enable/Disable abandoned/removed engine.

bMovObjEnabled

Enable/Disable surveillance engine .

bAdvTraEnabled

Enable/Disable advance tracker engine .

bCntLineEnabled

Enable/Disable Line counter engine.

bSmokeFireEnabled

Enable/Disable SmokeFire engine.

ulDetectionPoint

Detection point of tracked objects: 0:default/1:centroid/2:midbottom

reserved

Reserved for future use.

Remarks

The ulSecsToHoldOn time defines the amount of time that an object will be tracked by the engine once it becomes stationary. Since objects which become stationary must be "merged" into the scene after some finite time, the tracking engine will forget about objects that have become stationary after the ulSecsToHoldOn time . The default setting is 60 seconds.

The ulMinObjAreaPix defines the size of the smallest object that will be considered for tracking, in blobmap-pixels. Normally there is no need to modify this value. Decreasing it will allow the engine to track smaller objects at the expense of noise immunity.

To save CPU resource, enable or disable vca engines using bAbObjEnabled, bMovObjEnabled, bAdvTraEnabled, bCntLineEnabled and bSmokeFireEnabled.

VCA5_SCENECHANGE_INFO

Used to set the scene change detection parameters with the VCA5_CMD_SETSCENECHANGEPARAMS or VCA5_CMD_GETSCENECHANGEPARAMS command.

```
typedef struct _VCA5_SCENECHANGE_INFO{
    unsigned long      ulMode;
    VCA5_TAMPER_INFO   tTamper;
    unsigned long      ulDNColorEnabled;
    unsigned long      ulDNRimEnabled;
} VCA5_SCENECHANGE_INFO;
```

Members

ulMode

Select Scene change mode among DISABLE(0), AUTOMATIC(1) and MANUAL(2)

tTamper

Refer to [VCA5_TAMPER_INFO](#), when MANUAL mode is used.

ulDNColorEnabled

Enable/Disable the option to detect if color is changed by IR-cut filter.

ulDNRimEnabled

Enable/Disable the option to detect if Rim is moved by IR-cut filter.

Remarks

ulDNColorEnabled and ulDNRimEnabled are used to detect camera's IR-cut process to prevent false alarm, this IR-cut detection process works independently with ulMode. VCA detection is prone to working wrong if you set ulMode to DISABLE.

Mode defines

```
enum VCA_SCENECHANGE_MODE_E {
    VCA5_SCENECHANGE_MODE_DISABLED    = 0,
    VCA5_SCENECHANGE_MODE_AUTOMATIC  ,
    VCA5_SCENECHANGE_MODE_MANUAL
};
```

3.3. Rule type

A rule defines a condition, or set of conditions, that when met generates an event that can be output and used to alert another process to the fact. The types of rules available within VCAsys are as follows:

VCA5_RULE_TYPE_PRESENCE

Objects that are present inside a zone or pass over a line will trigger the rule and raise an alarm.

VCA5_RULE_TYPE_ENTER/ VCA5_RULE_TYPE_EXIT

An object entered alarm is raised when an object crosses from the outside to the inside of a detection zone or line. Conversely, an object exited alarm is raised when an object crosses from the inside to the outside of a detection zone or line:

VCA5_RULE_TYPE_APPEAR/ VCA5_RULE_TYPE_DISAPPEAR

An object appear alarm is raised when an object appears inside a detection zone. Note that this is different from object entered detection since the object must be initially detected inside the zone, e.g. people appearing in a doorway, or cars appearing from an underground car park.

Conversely an object disappear alarm is raised when an object disappears inside a detection zone. Again, this is different from object exit detection since the object must be tracked into the zone and then disappear without exiting the zone.

VCA5_RULE_TYPE_STOP

Objects that are stopped inside a zone for longer than the defined amount of time will trigger the rule and raise an alarm.

VCA5_RULE_TYPE_DWELL

Objects that dwell inside a zone for longer than the defined amount of time will trigger the rule and raise an alarm.

VCA5_RULE_TYPE_DIRECTION

Objects that travel in the configured direction (within the limits of the acceptance angle) through a zone or over a line trigger the rule and raise an alarm.

VCA5_RULE_TYPE_SPEED

Objects that travel within the bounds of the configured speeds, through a zone or over a line trigger the rule and raise an alarm.

VCA5_RULE_TYPE_TAILGATING

Object tailgating is defined as an object crossing a line or zone within a certain time after an object has already crossed the line or zone. If an object crosses a line or zone, and another object crosses the same line or zone within the specified time window, the Object Tailgating event will be triggered.

VCA5_RULE_TYPE_LINECOUNTER

A counting line is a detection filter optimized for bi-directional object counting (e.g. people or vehicles) in busier detection scenarios.

VCA5_RULE_TYPE_ABOBJ /VCA5_RULE_TYPE_RMOBJ

Objects that abandoned or removed inside a zone for longer than the defined amount of time will trigger the rule and raise an alarm.

VCA5_RULE_TYPE_ABOBJ and VCA5_RULE_TYPE_RMOBJ are same filter, this filter does not distinguish removed and abandoned object.

VCA5_RULE_TYPE_SMOKE/ VCA5_RULE_TYPE_FIRE

If fire or smoke is detected in the designated zone, it will trigger the rule and raise an alarm.

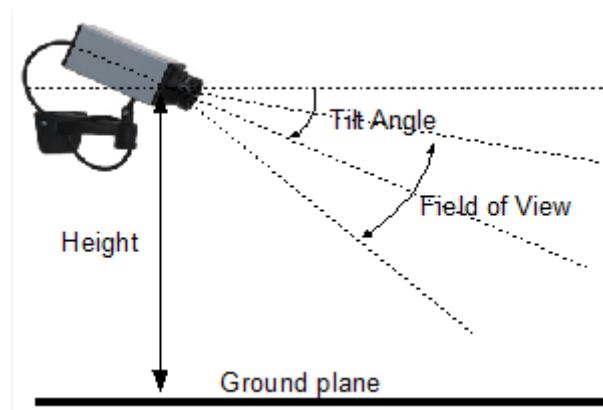
VCA5_RULE_TYPE_COLSIG

An alarm for an object color filter is raised when an object that contains more percentage than the specified threshold of the specified color is tracked inside the zone.

3.4. Camera calibration and object classification

3.4.1. Camera calibration

In order to estimate object parameters such as height, speed, area and classification, the camera must first be calibrated. Camera calibration involves establishing a reference to the ground plane in the scene.



3.4.2. Object classification

VCAsys can perform object classification once the camera has been calibrated. The object classification is based on properties extracted from the object including object area and speed. Objects that do not fit into any class are labeled as "Unclassified".

Revision History

MANUAL#	DATE(M/D/Y)	COMMENTS	Page Affected
01A.01	05/15/2009	Created	All
02A.04	08/20/2009	V1.0.3 updated & reviewed	All
03A.03	09/22/2009	V1.0.4 updated & reviewed	All
04A.04	12/29/2009	V1.0.5 updated & reviewed	All
05A.06	12/30/2009	V1.0.6 updated & reviewed	All
05A.07	01/26/2010	HP Series/NCP Series supported	7
06A.03	15/03/2010	V1.0.7 updated & reviewed	5, 16, 28, 29 40,
07A.00	03/23/2010	V1.0.8 updated & reviewed	4, 10
07A.01	04/02/2010	Corrected errors in Figure 1 and 2	5
07A.02	26/10/2010	Added updates for load/save XML config	All
07A.03	11/29/2010	Changed an incorrect term on 1.5. <i>Specification</i> (from <i>Objects</i> to <i>Classification groups</i>)	11
08A.02	07/11/2011	V1.2.0 updated & reviewed	All
08A.03	01/18/2012	V1.2.1 updated & reviewed	All
08A.04	07/27/2012	Resolution “1980” corrected to “1920”	13
09A.01	02/20/2013	V1.3.0 updated & reviewed	All
09B.02	4/June/2013	V1.4.0 updated	8, 51
02-2017-A	02/27/2017	Minor updates Manual name changed (UDA5 API Reference VCA5 API -> UDA5 VCA5 SDK Manual)	1