ApBase COM Development Guide

Documentation for the ApBase COM API

Introduction

The ApBase COM object makes key functions of the ApBase API available via the COM (Component Object Model) mechanism.

See the ApBase API Development Guide.

Quick Start

To use ApBase COM, create an object of type apbaseCom.ApBase. The details will depend on the tool or language you are using. The following example is in Matlab.

Here is a minimal procedure to open a demo kit camera, initialize the Aptina device, grab an image and convert it to RGB.

```
apbase = actxserver('apbaseCom.ApBase');
camera = ap.Create(0);
camera.LoadIniPreset('', '');
raw = camera.GrabFrame;
rgb = camera.ColorPipe(raw);
```

Note: The call to ap_LoadIniPreset() with blank parameters is only applicable in situations where the default initialization is sufficient. For demo kits with multiple possible configurations, such as ISP + sensor combinations or different connection options, you will need to specify the particular initialization preset for the configuration.

apbaseCom.ApBase Methods and Properties

First create an object of type abpaseCOM.ApBase. Then use this object to enumerate devices and create a camera object.

The ApBase object is free-threaded.

LONG DeviceProbe(BSTR sSensorDataDirOrFile)

Parameters

A file name or directory name, or an empty string.

Return Value

Error code.

Remarks

Search the system for Aptina demo kit devices or other ApBase compatible camera devices, identify the type of sensor or ISP, and build a set of internal data structures for each device. Use Create() to create a camera object.

If the parameter is an empty string, the default installation directory is searched for a sensor data file matching each device. Otherwise the parameter can be a specific sensor data file (.xsdat file) to be used with the device, or a directory to be searched for compatible sensor data files.

void CancelProbe()

Cancel a device probe in progress in another thread.

LONG NumCameras

Property Value

Number of cameras.

Remarks

Return the number of cameras discovered by the device probe function.

void Finalize()

Close all drivers and free all memory associated with the devices. Any existing camera objects become invalid. Called automatically when the object is released.

LONG LastError

Property Value

Error code.

Remarks

The error code from the last function call made in the current thread.

LONG LastSideEffects

Property Value

Side effects flags.

Remarks

The side effects flags from the last register write made in the current thread.

IApCamera Create(LONG nDeviceIndex)

Parameters

A device index, 0…NumCameras – 1.

Return Value

An apbaseCOM.ApCamera object.

Remarks

Create a camera object for one of the devices discovered by the device probe. If the device probe has not been done yet, this function will call DeviceProbe() with the default parameters.

Creating and destroying these objects does not affect the device or the device state information kept within ApBase.

After creating a camera device, call LoadIniPreset() or CheckSensorState() to make sure the state of the object is synchronized with the state of the device.

IApCamera CreateFromImageFile(BSTR sFilename)

Parameters

An image file name. Cannot be NULL.

Return Value

ApCamera object.

Remarks

Create a camera object from an image file. This is a way to make use of the image file reading feature of ApBase. Calling GrabFrame() on this object will retrieve the image. The ColorPipe() call can convert it to RGB if it is not RGB already, for example a Bayer image or YCbCr image.

Note: Although it’s possible to pass an image file name to DeviceProbe() and call Create(0) to get a camera device object, that method then precludes the possibility of probing for physical camera devices. This function is the preferred way to make a camera object from an image file.

IApCamera CreateVirtual(BSTR sFilename)

Parameters

A sensor data file name. Cannot be NULL.

Return Value

ApCamera object.

Remarks

Create an in-memory camera object from a sensor data file. This is a way to make use of the sensor data file parsing feature of ApBase. No I/O can be done on the resulting object, but it is possible to numerate the registers and get other property values defined in the sensor data file.

BSTR Home

Property Value

Path to the Aptina Imaging installation folder.

BSTR SensorData

Property Value

Path to the sensor_data installation folder where sensor data files are stored.

void OpenIoLog(ULONG nLogFlags, BSTR sFilename)

Parameters

nLogFlags – A combination of AP_LOG_x flags.

sFilename – Name of a file for the log data. A new file is created each time this function is called. Digits 0, 1, etc. will be appended to the file name.

Remarks

Starts a new log file. The file extension is usually .txt. Digits are appended to the filename so the previous log is not overwritten when the program is run again. Use ap_GetIoLogFilename() to get the actual log filename if needed.

void CloseIoLog()

Remarks

Close the current I/O log file.

BSTR IoLogFilename()

Property Value

The current I/O log file name.

Remarks

This function can be called after CloseIoLog(), and will return the most recent log filename.

void IoLogMsg(BSTR sText)

Parameters

sText – A text string to add to the log.

Remarks

The text will be added to the log if there is a log file currently open and AP_LOG was one of the flags passed to OpenIoLog(). A newline at the end of the string is not needed. The text will be preceded by a timestamp in the log file.

void IoLogDebugMsg(BSTR sText, BSTR sSource, BSTR sFunc, ULONG nLine)

Parameters

sText – The message for the log.

sSource – Name of the current source file. Can be empty.

sFunc – Name of the calling function.

nLine – Line number of the source file.

Remarks

The text will be added to the log if there is a log file currently open and AP_LOG_DEBUG was one of the flags passed to OpenIoLog(). A newline at the end of the string is not needed. The text will be preceded by a timestamp in the log file.

The sSource, sFunc, and nLine parameters are optional; if included they will be appended to the message. They are included to make it easy to find the location in your source code where a debug message originated from. Many compilers have macros like __FILE__, __FUNCTION__, __LINE__ that can be used here.

apbaseCom.ApCamera Properties

Properties of the apbaseCOM.ApCamera interface.

BSTR Name

Property Value

Name of device or sensor.

Remarks

Can be different from the part number. Read only.

BSTR PartNumber

Property Value

Part number.

LONG Version

Property Value

Revision number. Read only.

BSTR VersionName

Property Value

Version string.

Remarks

Usually like REV1, REV2, etc., but can be more descriptive in some cases. Read only.

BSTR FileName

Property Value

The name of the file the device parameters are based on.

Remarks

A .xsdat file for a physical camera, or an image or video file name. Read only.

LONG Width

Property Value

Image width in pixels.

Remarks

This property can be set, but setting it does not affect the state of a physical camera. The width, height, and image type can be set together in one call with SetImageFormat().

LONG Height

Property Value

Image height in pixels.

Remarks

This property can be set, but setting it does not affect the state of a physical camera. The width, height, and image type can be set together in one call with SetImageFormat().

BSTR ImageType

Property Value

The image type as a string.

Remarks

This property can be set, but setting it does not affect the state of a physical camera. The width, height, and image type can be set together in one call with SetImageFormat().

The possible image types are "BAYER-6", "BAYER-8", "BAYER-10", "BAYER-8+2", "BAYER-10IHDR", "BAYER-12", "BAYER-12HDR", "BAYER-8+4", "BAYER-14", "BAYER-14HDR", "BAYER-16", "BAYER-20", "BAYER-16+4", "BAYER-STEREO", "YCBCR", "YCBCR-16", "YCBCR-10", "YCBCR-20", "YUV-420", "M420", "RGB-565", "RGB-555", "RGB-444X", "RGB-X444", "RGB-332", "RGB-24", "RGB-32", "RGB-48", "BGRG", "JPEG", "JPEG-SPEEDTAGS", and "JPEG-ROT".

Functions for Executing Scripts

Functions of the apbaseCOM.ApCamera interface for initializing the camera.

LONG LoadIniPreset(BSTR sIniFile, BSTR sPreset)

Parameters

sIniFile – An ini file name. Can be NULL or empty.

sPreset – Name of the preset to run. Can be NULL or empty.

Return Value

Error code. 256 for success.

Remarks

Execute the ini file preset specified. If the sIniFile parameter is NULL or empty, the default ini file is used. If the sPreset parameter is NULL or empty, the default initialization procedure is executed, usually "Python:", if it exists, followed by "Demo Initialization".

After creating a camera device, call this function or CheckSensorState() to make sure the state of the object is synchronized with the state of the device.

LONG CheckSensorState(LONG nCheckFlags)

Parameters

Flags. Not used. Pass 0.

Return Value

Side effects flags.

Remarks

Re-read all registers on the device to determine the current modes, and update the internal data structures and the demo kit hardware.

Although ApBase examines all register writes as they happen to track the current modes of the camera, there are cases where the internal state of ApBase can get out of sync with the device. For example, downloading a firmware patch to an SOC or ISP can have results that can't be predicted by the software. This function can be used to re-sync ApBase to the device state.

Functions for Reading Sensor or ISP Registers

Functions of the apbaseCOM.ApCamera interface for accessing device registers.

ULONG GetSensorRegister(BSTR sRegisterName, BSTR sBitfieldName, LONG bCached)

Parameters

sRegisterName – Register name, from xsdat file.

sBitfieldName – Bitfield name, from xsdat file. Can be NULL or empty to read the whole register.

bCached – If non-zero, use a cached register value if available.

Return Value

Register value.

Remarks

Read the contents of a sensor or ISP register or bitfield, referenced by symbolic name.

ULONG GetSensorRegisterAddr(LONG nAddrType, ULONG nAddrSpace, ULONG nAddr, ULONG nDataBits, LONG bCached)

Parameters

nAddrType – Midlib address type symbol (MI_REG_ADDR, etc.)

nAddrSpace – Register page or addressable region.

nAddr – Register address.

nDataBits – Width of the register read operation.

bCached – If non-zero, use a cached value if available.

Return Value

Register value.

Remarks

Read the contents of a sensor or ISP register, referenced by register type and address.

Functions for Writing Sensor or ISP Registers

Functions of the apbaseCOM.ApCamera interface for accessing device registers.

Register write flags

Some register write functions can return status flags. The LastSideEffect property of the ApBase object can retrieve the flags from the most recent register write.

AP_FLAG_OK – No side-effect.

AP_FLAG_REALLOC – Image format (width, height) or buffer size may change.

AP_FLAG_PAUSE – Sensor may stop streaming.

AP_FLAG_RESUME – Sensor may resume streaming.

AP_FLAG_NOT_SUPPORTED – Selecting a mode that is not supported by the demo system.

AP_FLAG_ILLEGAL_REG_COMBO – The new value creates an invalid combination with some other register(s).

AP_FLAG_ILLEGAL_REG_VALUE – The new value is not supported by the device.

AP_FLAG_REGISTER_RESET – Many other register values will change (reset or state change).

AP_FLAG_CLOCK_FREQUENCY – The clock frequency will change (this is a PLL register or clock divider).

AP_FLAG_REG_LIST_CHANGED – The set of camera registers will change.

AP_FLAG_NOT_ACCESSIBLE – Register is not accessible (standby).

AP_FLAG_READONLY – Writing a read-only register.

AP_FLAG_WRITEONLY – Reading a write-only register.

LONG SetSensorRegister(BSTR sRegisterName, BSTR sBitfieldName, ULONG nValue)

Parameters

szRegisterName – Register name, from xsdat file.

szBitfieldName – Bitfield name, from xsdat file. Can be NULL or empty to write the whole register.

nValue – New register value.

Return Value

Side effects flags.

Remarks

Write the contents of a sensor or ISP register or bitfield, referenced by symbolic name.

LONG SetSensorRegisterAddr(LONG nAddrType, ULONG nAddrSpace, ULONG nAddr, ULONG nDataBits, ULONG nValue)

Parameters

nAddrType – Midlib address type symbol (MI_REG_ADDR, etc.)

nAddrSpace – Register page or addressable region.

nAddr – Register address.

nDataBits – Width of the register read operation.

nValue – New value for the register.

Return Value

Side effects flags.

Remarks

Same as SetSensorRegister(), but the register is referenced by register type and address, and always writes the whole register.

Functions for Getting and Processing Images

Functions of the apbaseCOM.ApCamera interface for getting images.

LONG SetImageFormat(ULONG nWidth, ULONG nHeight, BSTR sImageType)

Parameters

nWidth – New image width, or 0 to keep the current width.

nHeight – New image height, or 0 to keep the current height.

szImageType – Name of new image type, or NULL to keep the current image type.

Return Value

Error code.

Remarks

ApBase normally calculates the image format from the device register settings automatically. This function can be used as an override if necessary.

VARIANT GrabFrame()

Return Value

A 1-dimensional array of pixel values or byte values.

Remarks

Get the next image from the camera.

Use LastError for the error code.

VARIANT ColorPipe(VARIANT inImage, ULONG *outWidth, ULONG *outHeight, ULONG *outBitDepth)

Parameters

inImage – An image received from the device, or a similarly formatted image.

outWidth – Pointer to a memory location to receive the width of the output image. May be NULL.

outHeight – Pointer to a memory location to receive the height of the output image. May be NULL.

outBitDepth – Pointer to a memory location to receive the pixel size of the output image. May be NULL.

Return Value

The processed image, normally an array of RGB pixel values.

Remarks

Convert image data returned from GrabFrame(), or similarly formatted image, to RGB. The output is normally 32 bits per pixel with B – G – R – X byte order.

The converted image could conceivably have different dimensions than the input image. The rgbWidth, rgbHeight and rgbBitDepth parameters will be filled in with the converted image size. These parameters may be NULL.

LONG GetState(BSTR sState)

Parameters

sState – Name of an integer-valued color pipe state variable.

Return Value

Value of the variable.

Remarks

Get the value of an integer-valued color pipe state variable. See the INI File User’s Guide for a list of color pipe variables.

void SetState(BSTR sState, LONG nValue)

Parameters

sState – Name of an integer-valued color pipe state variable.

nValue – New value for the variable.

Remarks

Set the value of an integer-valued color pipe state variable.

BSTR GetStateStr(BSTR sState)

Parameters

sState – Name of a string-valued color pipe state variable.

Return Value

Value of the variable.

Remarks

Get the value of a string-valued color pipe state variable.

void SetStateStr(BSTR sState, BSTR sValue)

Parameters

sState – Name of an string-valued color pipe state variable.

sValue – New value for the variable.

Remarks

Set the value of a string-valued color pipe state variable.

Functions for Enumerating Sensor or ISP Registers

Methods of the apbaseCOM.ApCamera interface for enumerating device registers.

LONG NumRegisters(BSTR sDevice)

Parameters

sDevice – Name of the chip. It can be a sensor or ISP part number, or a demo kit component. An empty string will refer to the sensor or ISP.

Return Value

Number of registers defined for the specified device.

Remarks

The return value minus 1 is the maximum value that can be used as the nIndex parameter of ApCamera.Register(), below.

IApRegister Register(BSTR sDevice, LONG nIndex)

Parameters

sDevice – Name of the chip. It can be a sensor or ISP part number, or a demo kit component. An empty string will refer to the sensor or ISP.

nIndex – A value from 0 to NumRegisters() – 1 that can be used to get each of the defined registers in turn.

Return Value

An ApRegister object that is a reference to the specified register.

Remarks

.

IApRegister FindRegister(BSTR sDevice, BSTR sRegister)

Parameters

sDevice – Name of the chip. It can be a sensor or ISP part number, or a demo kit component. An empty string will refer to the sensor or ISP.

sRegister – The symbolic name of the register.

Return Value

An ApRegister object that is a reference to the specified register.

Remarks

See apbaseCom.ApRegister Properties below.

IApBitfield FindBitfield(BSTR sDevice, BSTR sRegister, BSTR sBitfield)

Parameters

sDevice – Name of the chip. It can be a sensor or ISP part number, or a demo kit component. An empty string will refer to the sensor or ISP.

sRegister – The symbolic name of the register.

sBitfield – The symbolic name of the bitfield.

Return Value

An ApBitfield object that is a reference to the specified bitfield.

Remarks

See apbaseCom.ApBitfield Properties below.

apbaseCom.ApRegister Methods

Methods of the apbaseCOM.ApRegister interface.

IApBitfield Bitfield(LONG nIndex)

Parameters

nIndex – A value from 0 to NumBitfields() – 1 that can be used to get each of the defined bitfields in turn.

Return Value

An ApBitfield object that is a reference to the specified bitfield.

Remarks

See apbaseCom.ApBitfield Properties below.

IApBitfield BitfieldByName(BSTR sBitfield)

Parameters

sBitfield – The symbolic name of the bitfield.

Return Value

An ApBitfield object that is a reference to the specified bitfield.

Remarks

See apbaseCom.ApBitfield Properties below.

apbaseCom.ApRegister Properties

Properties of the apbaseCOM.ApRegister interface.

BSTR Symbol

Property Value

Symbolic name of the register.

Remarks

This name is used by other API calls, in INI file scripts, etc.

BSTR DisplayName

Property Value

Display name of the register.

Remarks

Very often, this is the same as the symbolic name, but in lower case.

BSTR Detail

Property Value

Short description of the register.

Remarks

A one-line description of the register.

BSTR LongDesc

Property Value

Long description of the register.

Remarks

This is the full data sheet documentation of the register.

ULONG Bitwidth

Property Value

The width of the register in bits.

Remarks

Can be 8, 16 or 32. In many cases, only a subset of the bits are meaningful.

ULONG Mask

Property Value

Which bits of the register are meaningful.

Remarks

.

ULONG Default

Property Value

The default value of the register.

Remarks

.

BSTR Datatype

Property Value

Data type of the register.

Remarks

Example values are “unsigned”, “signed”, “fixed8”, etc.

DOUBLE Minimum

Property Value

The minimum recommended or useful value of the register.

Remarks

The return value is converted to floating point according to the data type of the register.

DOUBLE Maximum

Property Value

The maximum recommended or useful value of the register.

Remarks

The return value is converted to floating point according to the data type of the register.

ULONG Addr

Property Value

The register address.

Remarks

.

ULONG AddrSpace

Property Value

The address space of the register.

Remarks

.

BSTR AddrSpaceId

Property Value

Symbolic name of the register's address space.

Remarks

.

BSTR AddrSpaceName

Property Value

Display name of the register's address space.

Remarks

.

LONG AddrType

Property Value

The address type of the register.

Remarks

Possible values are 0 for simple register, 1 for firmware variable, 2 for advanced register other RAM address, 3 for other indirectly addressed register, or 4 for attached sensor register (as on an ISP with an attached sensor).

ULONG Rw

Property Value

The allowed access to the register.

Remarks

Possible values are 0 for read/write, 1 for read-only or 2 for write-only.

apbaseCom.ApBitfield Properties

Properties of the apbaseCOM.ApBitfield interface.

BSTR Symbol

Property Value

Symbolic name of the bitfield.

Remarks

This name is used by other API calls, in INI file scripts, etc.

BSTR DisplayName

Property Value

Display name of the bitfield.

Remarks

Very often, this is the same as the symbolic name, but in lower case.

BSTR Detail

Property Value

Short description of the bitfield.

Remarks

A one-line description of the bitfield.

BSTR LongDesc

Property Value

Long description of the bitfield.

Remarks

This is the full data sheet documentation of the bitfield.

ULONG Bitwidth

Property Value

The width of the bitfield in bits.

Remarks

Can be any value from 1 to the width of the parent register.

ULONG Mask

Property Value

Which bits of the parent register belong to this bitfield.

Remarks

.

ULONG AdjustedMask

Property Value

The bitfield bitmask shifted right to the LSB.

Remarks

This mask is suitable for masking off a value to be written to the bitfield.

ULONG Default

Property Value

The default value of the bitfield.

Remarks

.

BSTR Datatype

Property Value

Data type of the bitfield.

Remarks

Example values are “unsigned”, “signed”, “fixed8”, etc.

DOUBLE Minimum

Property Value

The minimum recommended or useful value of the bit bitfield mask.

Remarks

The return value is converted to floating point according to the data type of the bitfield.

DOUBLE Maximum

Property Value

The maximum recommended or useful value of the bitfield.

Remarks

The return value is converted to floating point according to the data type of the bitfield.

ULONG Rw

Property Value

The allowed access to the bitfield.

Remarks

Possible values are 0 for read/write, 1 for read-only or 2 for write-only.