VPMR v.7

DLL Reference Manual

August 16

Table of Contents

Initialization / Finalization

vpmrInit

Initializes the **Vehicle Plates Automatic Reader** (**VPAR**). It loads the Artificial Neural Networks used by the OCR and initializes parameters. This function must be called before calling any other function in this library.

```
long vpmrInit (   char* PathtoMap,
                  long lCountryCode,
                  long lAverageCharacterHeight,
                   bool bDuplicateLines,
                   bool lreserved1,
                   long lreserved2,
                   bool bTrace = false  );
```

Arguments

PathToMap	Path to folders that contains map files
lCountryCode	County code used for selecting the target country for license *plate recognition. See declaration file for a list of supported countries*.
lAvCharacterHeight	Approximate average height of the characters in the plates to read. If this argument is **-1**, the library uses *automatic height mode* and tries to read characters of any height. If **-1** is passed, the processing time will be increased.
bDuplicateLines	In order to properly recognize images acquired with only half of the scan lines, this argument must be **true**. For images acquired with all the lines, this parameter must be **false**.
lreserved1	Sort characters in squared plates (plates with two rows of characters). If this argument is **false**, the characters in the top row are returned first, followed by the characters in the bottom row. If it is **true**, the characters are re-arranged to match the Spanish format. (For example, if this parameter is **true**, a plate with the top line "BU AX" and bottom line "5278" would be re-arranged to generate the result "BU5278AX". In the other hand, if this argument is **false**, the result would be "BUAX5278").

lreserved2 Activate special filter for colour treatment. Possible values are:

0	Average value of the three channels (**Recommended, default value**)
1	Use first colour channel (red for RGB image or blue in case of BGR)
2	Use second colour channel (green always)
3	Use the third colour channel (blue for RGB image, red for BGR image)
< 0	Error
> 3	Error

bTrace This parameter must be set to **false**.

Return Value

0 → Error.
1 → Ok.

vpmrEnd

Frees the memory allocated by the **Vehicle Plates Automatic Reader (VPMR)**. Only call this function at the end of the program.

void **vpmrEnd** (void);

vpmrAddCountry

This feature should only be used when the library is initialized in multicountry **(code 100)** mode. In this case, AddCountry is used to add countries to consider for reading. Initially (after the Init), no country is included and must be called at least once.

long **vpmrAddCountry** (long *lCountryCode*);

Parámetros

lCountryCode Country code.

Return Value

0 → Error.
1 → Ok.

vpmrRead

This function reads the license plate present within an image. The input to this function is an image. It analyzes the image looking for a vehicle plate and if it finds it, it reads the plate. A set of functions is supplied to retrieve the result of the recognition (see *Retrieving Results* chapter bellow).

The input supplied to this function is the *image buffer* in **256 grayscale** levels (1 byte per pixel). The *width* and *height* of the image must be supplied as well.

```
long vpmrRead ( long lWidth,
                long lHeight,
                unsigned char * pbImageData );
```

Arguments

lWidth	Width (in pixels) of the image that will be analyzed.
lHeight	Height (in pixels) of the image that will be analyzed.
pbImageData	*Buffer* with the image data (pixels), in 256 grey levels (1 *byte* per *pixel*).

Return Value

0 → Error.
1 → Ok.

vpmrReadRGB24

This function reads the vehicle plate present within an image. The input to this function is an image. It analyzes the image looking for a vehicle plate and if it finds it, it reads the plate. A set of functions is supplied to retrieve the result of the recognition (see *Retrieving Results* chapter bellow).

The input supplied to this function is the *image buffer* in **RGB-24 bits** (3 bytes per pixel) format. The *width* and *height* of the image must be supplied as well.

```
long vpmrReadRGB24 ( long lWidth,
                     long lHeight,
                     unsigned char * pbImageData,
                     bool bFlip = false );
```

Arguments

lWidth	Width (in pixels) of the image that will be analyzed.
lHeight	Height (in pixels) of the image that will be analyzed.
pbImageData	*Buffer* with the image data (pixels) using 3 bytes per pixel (RED, GREEN, BLUE).
bFlip	This value must be **true** only if the RGB buffer contains first the bottom row of the image, then the next one upwards, and so on. The last line of values in the buffer contains the top row of pixels in the image. Some devices acquire the RGB buffer in this way (*bottom-up*).

Return Value

0 → Error.
1 → Ok.

vpmrReadRGB32

This function reads the vehicle plate present within an image. The input to this function is an image. It analyzes the image looking for a vehicle plate and if it finds it, it reads the plate. A set of functions is supplied to retrieve the result of the recognition (see *Retrieving Results* chapter bellow).

The input supplied to this function is the *image buffer* in **RGB-32 bits** (4 bytes per pixel) format. The *width* and *height* of the image must be supplied as well.

```
long vpmrReadRGB32 ( long lWidth,
                     long lHeight,
                     unsigned char * pbImageData,
                     bool bFlip = false );
```

Arguments

lWidth	Width (in pixels) of the image that will be analyzed.
lHeight	Height (in pixels) of the image that will be analyzed.
pbImageData	*Buffer* with the image data (pixels) using 4 bytes per pixel (RED, GREEN, BLUE, ALPHA).
bFlip	This value must be **true** only if the RGB buffer contains first the bottom row of the image, then the next one upwards, and so on. The last line of values in the buffer contains the top row of pixels in the image. Some devices acquire the RGB buffer in this way (*bottom-up*).

Return Value

0 → Error.
1 → Ok.

vpmrReadBMP

This function reads the vehicle plate present within an image. The input to this function is an image. It analyzes the image looking for a vehicle plate and if it finds it, it reads the plate. A set of functions is supplied to retrieve the result of the recognition (see *Retrieving Results* chapter bellow).

The input supplied to this function is an image file in standard Bitmap (BMP) format.

long **vpmrReadBMP** (char * *strFilename*);

Arguments

strFilename	Filename of BMP image to process.

Return Value

0 → Error.
1 → Ok.

vpmrReadJPG

This function reads the vehicle plate present within an image. The input to this function is an image. It analyzes the image looking for a vehicle plate and if it finds it, it reads the plate. A set of functions is supplied to retrieve the result of the recognition (see *Retrieving Results* chapter bellow).

The input supplied to this function is an image file in standard Jpeg (JPG) format.

long **vpmrReadJPG** (char * *strFilename*);

Arguments

strFilename	Filename of JPG image to process.

Return Value

0 → Error.
1 → Ok.

Retrieving Results

vpmrGetNumberOfPlates

Returns the number of plates found and recognized in the last analyzed image.

long **vpmrGetNumberOfPlates** (void);

Return Value

Returns the number of vehicle plates read.

Remarks

This value will be **0** if no vehicle plate is found or if it cannot be read. It will be **1** if one vehicle plate was found and recognized and **2** only if ***vpmrInit*** was called with the argument *bTrailersOn* set to **true** and the processed image contains the two plates at the back of a truck carrying a trailer.

vpmrGetText

Returns the text (in **ASCII** format) of the vehicle plate read in the last processed image.

long **vpmrGetText** (char * *strResult*,
long *lPlate* = 0);

Arguments

strResult	String where the ASCII text is returned.
lPlate	Index of the plate we want to retrieve. If only one plate was read, number **0** must be specified. If two plates where read (a truck with two plates at the back), the result for plate number **0** or number **1** can be requested.

Return Value

0 → Error.
1 → Ok.

vpmrGetNumberOfCharacters

Returns the number of characters present in the last plate processed by **VPAR**.

long **vpmrGetNumberOfCharacters** (long *lPlate* = 0);

Arguments

lPlate	Index of the plate we want to retrieve. If only one plate was read, number **0** must be specified. If two plates where read (a truck with two plates at the back), the result for plate number **0** or number **1** can be requested.

Return Value

Number of recognized characters.

vpmrGetPolarity

Returns the polarity of the plate.

long **vpmrGetPolarity** (long *lPlate* = 0);

Arguments

lPlate	Index of the plate we want to retrieve. If only one plate was read, number **0** must be specified. If two plates where read (a truck with two plates at the back), the result for plate number **0** or number **1** can be requested.

Return Value

Polarity of the plate: 1 for dark chars on bright backgorund, 2 otherwise.

vpmrGetGlobalConfidence

Returns a confidence factor for the result of the last plate recognized by the **Vehicle Plates Automatic Reader**.

This value is expressed as a percentage (0% – 100%).

float **vpmrGetGlobalConfidence** (long *lPlate* = 0);

Arguments

lPlate	Index of the plate we want to retrieve. If only one plate was read, number **0** must be specified. If two plates where read (a truck with two plates at the back), the result for plate number **0** or number **1** can be requested.

Return Value

The Confidence Factor for the last recognition.

vpmrGetAverageCharacterHeight

Returns the average height (in pixels) of the characters present in the last plate recognized by the **Vehicle Plates Automatic Reader**.

float **vpmrGetAverageCharacterHeight** (long *lPlate* = 0);

Arguments

lPlate	Index of the plate we want to retrieve. If only one plate was read, number **0** must be specified. If two plates where read (a truck with two plates at the back), the result for plate number **0** or number **1** can be requested.

Return Value

The average height (in pixels) of the characters in the last plate analyzed.

vpmrGetCharacterConfidence

Returns a confidence factor for a given character within the last plate analyzed.

This value is expressed as a percentage (0% – 100%).

```
float vpmrGetCharacterConfidence ( long lIndex ,
                                   long lPlate = 0 );
```

Arguments

lIndex	(0..n) Index of the character we want to retrieve the confidence factor for.
lPlate	Index of the plate we want to retrieve. If only one plate was read, number **0** must be specified. If two plates where read (a truck with two plates at the back), the result for plate number **0** or number **1** can be requested.

Return Value

Character Confidence Factor.

vpmrGetRectangle

Returns the coordinates of the rectangle containing the vehicle license plate in the last image processed.

```
void vpmrGetRectangle ( long * plLeft,
                        long * plTop,
                        long * plRight,
                        long * plBottom,
                        long lPlate = 0 );
```

Arguments

plLeft	X-Coordinate of the Upper-Left corner of the rectangle.
plTop	Y-Coordinate of the Upper-Left corner of the rectangle.
plRight	X-Coordinate of the Lower-Right corner of the rectangle.
plBottom	Y-Coordinate of the Lower-Right corner of the rectangle.
lPlate	Index of the plate we want to retrieve. If only one plate was read, number **0** must be specified. If two plates where read (a truck with two plates at the back), the result for plate number **0** or number **1** can be requested.

vpmrGetFormat

This property returns the country code that matches license plate result. Example: If the software detects reading of a Spanish license plate, code returned is 101. If no country is detected it returns 0. For European license plates (no country detected) it returns 100.

long **vpmrGetFormat** (long *lPlate*);

Parameters

lPlate Number of license plate read result to query about (0 to 7).

Return value

0 → No license plate format detected.
N → Country code or continent of detected format.

vpmrGetCharRectangle

Returns the coordinates of the rectangle containing a given character in the last plate analyzed. **The coordinates are relatives to saved image by vpmrSavePlateImage function.**

```
void vpmrGetRectangle (long lIndex,
                       long * plLeft,
                       long * plTop,
                       long * plRight,
                       long * plBottom,
                       long lPlate = 0 );
```

Parameters

lIndex	Index of the character (first character has index = 0).
plLeft	X-Coordinate of the Upper-Left corner of the rectangle.
plTop	Y-Coordinate of the Upper-Left corner of the rectangle.
plRight	X-Coordinate of the Lower-Right corner of the rectangle.
plBottom	Y-Coordinate of the Lower-Right corner of the rectangle.
lPlate	If there is more than one plate found, this is the plate index (0 for the first one)

Time Management

vpmrGetProcessingTime

Returns the processing time for the last reading operation.

This value is expressed in milliseconds.

long **vpmrGetProcessingTime** ( );

Return Value

Processing time (in milliseconds) for the last reading.

vpmrSetTimeOut

This function specifies the maximum processing time for reading operations.

This value is expressed in milliseconds.

void **vpmrSetTimeOut** (long *lMilliseconds*);

Arguments

lMilliseconds	Maximum processing time (in milliseconds).

Remarks

This maximum time is approximate. This means that the actual processing time can be (in some cases) slightly longer that the time-out specified.

Optional Configuration

vpmrSetCorrectionCoefficients

This function sets the distortion correction coefficients that will be applied to all the images before being analyzed.

The types of distortions that can be corrected are: tangential distortion (horizontal and/or vertical perspective) and rotation.

The specified coefficients will be applied to all the images until this function is called again with different arguments of ***vpmrSetDistortionCorrectionOff*** is called.

```
void vpmrSetCorrectionCoefficients (    float fDistance,
                                        float fVerticalCoeff,
                                        float fHorizontalCoeff,
                                        float fAngle);
```

Arguments

fDistance	Approximate distance between camera and object (in meters).
fVerticalCoeff	Coefficient for correcting Vertical Perspective.
fHorizontalCoeff	Coefficient for correcting Horizontal Perspective.
fAngle	Angle for correcting Rotation.

vpmrSetCorrectionCoefficientsEx

This function sets the distortion correction coefficients that will be applied to all the images before being analyzed.

The types of distortions that can be corrected are: tangential distortion (horizontal and/or vertical perspective) and rotation.

The specified coefficients will be applied to all the images until this function is called again with different arguments of ***vpmrSetDistortionCorrectionOff*** is called.

```
void vpmrSetCorrectionCoefficients (    float fDistance,
                                        float fVerticalCoeff,
                                        float fHorizontalCoeff,
                                        float fAngle,
                                        float fVerticalSkew,
                                        float fHorizontalSkew
                                        );
```

Arguments

fDistance	Approximate distance between camera and object (in meters).
fVerticalCoeff	Coefficient for correcting Vertical Perspective.
fHorizontalCoeff	Coefficient for correcting Horizontal Perspective.
fAngle	Angle for correcting Rotation.
fVerticalCoeff	Coefficient for correcting Vertical Skew.
fHorizontalCoeff	Coefficient for correcting Horizontal Skew.

vpmrSetDistortionCorrectionOff

This function deactivates the distortion correction pre-process.

Use ***vpmrSetCorrectionCoefficients*** to activate it again.

void **vpmrSetDistortionCorrectionOff** ();

vpmrConfigureAutomaticCharacterHeight

This function configures the Automatic Character Height steps.

When ***vpmrInit*** is called with the argument *lAvCharacterHeight* set to -1, the automatic character height mode is selected. By default, the range of character heights scanned in this mode is from 25 pixels to 60 pixels.

By using this function, the user can select the heights that VPMR will scan.

long **vpmrConfigureAutomaticCharacterHeight** (long *lNumSteps*,
long * *plSteps*);

Arguments

lNumSteps	Always 2.
plSteps	Array with the start of the interval in the first position and the end of the interval in second position, e.g. 25, 45

Return Value

0 → Error.
1 → Ok.

Remarks

To recover the default configuration, the following code must be executed:

```
long lSteps[2] = { 25, 60 };
vpmrConfigureAutomaticCharacterHeight (2, lSteps );
```

vpmrSetRectangle

Sets the rectangle within the image where the inspection process will take place. Only the provided region will be inspected to look for a license plate.

This function can be used to speed up the process when the approximate position of the license plate within the image is known.

```
long vpmrSetRectangle (     long lLeft,
                            long lTop,
                            long lWidth,
                            long lHeight );
```

Arguments

lLeft, lTop	Left-top corner coordinates of the rectangle (in *píxels*).
lWidth, lHeight	Rectangle dimensions (in *píxels*).

Return Value

0 → Error.
1 → Ok.

Remarks

The inspection rectangle specified will be applied to all the further inspections until *vpmrSetRectangle* is called again with different parameters.
In order to inspect the whole image, vpmrSetRectangle must be called with all its parameters set to 0:

vpmrSetRectangle (0, 0, 0, 0);

By default, after library initialization with *vpmrInit*, the inspection rectangle is set to inspect the whole image.

vpmrShadowKillerOn

Activates the pre-processing for dealing with shadows cast on the upper side of the plates.

void **vpmrShadowKillerOn** ();

vpmrShadowKillerOff

Deactivates the pre-processing for dealing with shadows cast on the upper side of the plates.

void **vpmrShadowKillerOff** ();

vpmrStrictSyntaxOn

By calling this function, the format returned for custom plates (plates not following the standard country's syntax) is 0.

void **vpmrStrictSyntaxOn** ();

vpmrStrictSyntaxOff

By calling this function, the format returned for custom plates (plates not following the standard country's syntax) is the country's format (this is the default mode).

void **vpmrStrictSyntaxOff** ();

vpmrSetScaleFactor

Scales the image internally before processing it.

void **vpmrSetScaleFactor** (float *fScale*);

Arguments

fScale	Scale factor. Width and height dimensions of the input image are multiplied by this factor.

Remark: The scaling takes place internally and therefore the configuration parameters (character height, rectangle of interest, etc...) must relate to the original image.
The results (average character height, rectangle, etc...) also relate to the original dimensions of the image.

vpmrReturnSpaces

By calling this function after vpmrInit the number plates recognized will be returned including the spaces between characters.

void **vpmrReturnSpaces** ();

Storing Data into the HASP Dongle

vpmrWriteHASP

Writes data into the internal memory of the HASP dongle. This capability for storing data into the HASP can be used for any purpose. Data is encrypted automatically before being written into the HASP memory.

A maximum of 24 bytes can be written.

long **vpmrWriteHASP** (unsigned char * *pData,*
long *lSize*);

Arguments

pData	Buffer containing the data to be written in to the HASP internal memory.
lSize	Size (in bytes) of the data to write (maximum 24 bytes).

Return Value

0 → Error.
1 → Ok.

vpmrReadHASP

Reads the data stored in the internal memory of the HASP dongle. Data is automatically decrypted after being read from the HASP memory.

long **vpmrReadHASP** (unsigned char * *pData,*
long *lSize*);

Arguments

pData	Buffer where the retrieved data will be stored.
lSize	Size (in bytes) of the data to read.

Return Value

0 → Error.
1 → Ok.

Example of Use

```
void func ()
{
    bool ok;
    // The characters in the images to process have about 30 pixels in height,
    // there is no need to duplicate horizontal lines (all the scan lines are acquired)
    // and we want two-lines plates to be re-arranged to fit the spanish format.
    ok = (bool) vpmrInit (PATH,_CODE_ITA,30, false, true, 0, false);
    if (ok)
        {
        unsigned char * buffer;   // Image buffer.
        long numchars;            // Number of characters read.
        char text[32];            // String where the result will be stored.
        float Cf;                 // Confidence Factor of the reading result.
        float characterCf[32];    // Array to store the characters confidence factor.
        // Buffer memory allocation.
        . . .
        // Acquire the image buffer in 256 grayscale levels (1 byte per pixel) and store it
        // into buffer.
        . . .
        // Image dimensions are 384x288 pixels.
        ok = (bool) vpmrRead (384, 288, buffer);
        if (ok)
            {
            vpmrGetText (text, 0);
            Cf = vpmrGetGlobalConfidence (0);
            numchars = vpmrGetNumberOfCharacters (0);
            for (long i = 0; i < numchars; i++)
                {
                characterCf[i] = vpmrGetCharacterConfidence (i, 0);
                }
            }
        else
            {
            // Error reading license plate.
            }

            // Read more images.
        . . .
            // Finalize the software

      vpmrEnd ();
      }
    else
      {
      // Error initializing the VPMR.
      }
}
```