VPAR v.7

DLL Reference Manual

28 February 2019

Table of Contents

Initialization / Finalization

vpmrInitialize

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 vpmrInitialize ( long lCountryCode,
                      bool bTrace = false  );
```

Arguments

lCountryCode	County/state code used for selecting the target country for license plate recognition. *(See **Appendix 1** for a list of supported countries/states).*
bTrace	This parameter must be set to **false**.

Return Value

0 → Error.
1 → Ok.

vpmrInit

This is an extended version of the previous function Initialize.

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 ( long lCountryCode,
                long lAverageCharacterHeight,
                bool bDuplicateLines = false,
                long lreserved1,
                long lreserver2,
                bool bTrace = false  );
```

Arguments

lCountryCode	County code used for selecting the target country for license plate recognition. *(See **Appendix 1** for a list of supported countries/states).*
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**.
bTrace	This parameter must be set to **false** unless required otherwise by a Neural Labs technician.

Return Value

0 → Error.
1 → Ok.

vpmrInitEx

This is an extended version of the previous function Init.

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. This function allows to specify the geographic region, overriding the region specified in the .ini files).

```
long vpmrInitEx ( long lRegion,
                  long lCountryCode,
                  long lAverageCharacterHeight,
                  bool bDuplicateLines = false,
                  long lreserved1,
                  long lreserver2,
                  bool bTrace = false  );
```

Arguments

lRegion	Geographic region where the software will be installed. The valid codes are: **1** = Europe **2** = South America **3** = North America **4** = Asia **5** = Africa **6** = Oceania
lCountryCode	County code used for selecting the target country for license plate recognition.
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.
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**.
bTrace	This parameter must be set to **false**.

Return Value

0 → Error.
1 → Ok.

vpmrAddCountry

This function is used when the library has been initialized with the country code LOC_CODE_MULTI. In this case, AddCountry is used to add countries to take into account for reading.

If the initialization function was called with country code = LOC_CODE_MULTI, no countries are included initially and AddCountry must be called at least once.

*(See **Appendix 1** for a list of supported countries/states).*

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

Parameters

lCountryCode Country code to add.

Return value

0 → Error.
1 → Ok.

vpmrEnd

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

void **vpmrEnd** (void);

Engine Configuration

vpmrSetCorrectionCoefficients

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

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*** or ***vpmrSetCorrectionCoefficientsEx*** to activate it again.

void **vpmrSetDistortionCorrectionOff** ();

vpmrConfigureCharacterHeightRange

This function configures the Automatic Character Height range.

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 15 pixels to 80 pixels.

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

long **vpmrConfigureCharacterHeightRange** (long *lMinHeight*,
long *lMaxHeight*);

Arguments

lMinHeight	Minimum character height (in pixels).
lMaxHeight	Maximum character height (in pixels).

Return Value

0 → Error.
1 → Ok.

vpmrSetScaleFactor

Scales the image internally before processing it.

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

Parámetros

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

***Remarks*:**

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.

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.

vpmrSetPolarityStrategy

Specifies the strategy for dealing with plate polarity. If this function is never called, the polarity strategy used dependes on the country or countries initialized.

void **vpmrSetPolarityStrategy** (long *lPolarityMode*);

Arguments

lPolarityMode Polarity strategy. It must be on of these values:

POLARITY_SEARCH_DO_NOT_INVERT (1):
Only searches for plates with dark characters on brighter background.

POLARITY_SEARCH_INVERT (2):
Only searches for plates with bright characters on darker background.

POLARITY_SEARCH_INVERT_IF_FAILS (3):
Searches for plates with dark characters on brighter background and if no plate is recognized, then searches for plates with bright characters on darker background.

POLARITY_SEARCH_DO_NOT_INVERT_IF_FAILS (4):
Searches for plates with bright characters on darker background and if no plate is recognized, then searches for plates with dark characters on brighter background.

POLARITY_SEARCH_BOTH (5):
Always searches for plates with any polarity.

vpmrConfigurePlateTypes

Specifies if the library must search for plates with one line, two lines or three lines.

void **vpmrConfigurePlateTypes** (bool *zSearch1Line,*
bool *zSearch2Lines,*
bool *zSearch3Lines*);

Arguments

zSearch1Line	This argument must be true in order to search plates with 1 line. false otherwise.
zSearch2Lines	This argument must be true in order to search plates with 2 lines. false otherwise.
zSearch3Lines	This argument must be true in order to search plates with 3 lines. false otherwise.

vpmrShadowKillerOn

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

void **vpmrShadowKillerOn** (long *lLevel* = 1);

Arguments

lLevel This parameter can have the following values:
0 = Disabled
1 = Standard shadow killer level
2 = Advanced shadow killer level (to be used only if level 1 is not enough)

vpmrShadowKillerOff

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

void **vpmrShadowKillerOff** ();

vpmrSlantCorrectionOn

Activates the automatic slant detection and correction.

void **vpmrSlantCorrectionOn** ( );

Remarks:

This is the working mode by default.

vpmrSlantCorrectionOff

Deactivates the automatic slant detection and correction.

void **vpmrSlantCorrectionOff** ();

vpmrReturnSpacesOn

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

void **vpmrReturnSpacesOn** ( );

vpmrReturnSpacesOff

By calling this function after Init the number plates recognized will be returned without the spaces between characters.

void **vpmrReturnSpacesOff** ( );

Remarks:

This is the default setting.

Reading Plates

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 below).

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.

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 below).

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

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

Arguments

strFilename	Filename of a standard BMP file (24 bits/pixel)

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 below).

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

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

Arguments

strFilename	Filename of a standard JPG file.

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 below).

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 below).

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.

vpmrReadPD

This function reads the license plate present within an image using presence detection to determine if anything has changed from the previous frame.
The Presence Detection Check can help to save a lot of processing time when there aren't changes in the scene.

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 below).

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 vpmrReadPD (  long lWidth,
                   long lHeight,
                   unsigned char* pbImageData,
                   long lMode,
                   long lThreshold,
                   unsigned char* pbImageDataRef );
```

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.
lMode	This is the operating mode for the Presence Detection. The valid values are: **0** --> Presence detection disabled. **1** --> Presence detection only. **2** --> Presence detection + ROI. **3** --> Presence detection + ROI + Mask.
lThreshold	Sensitivity threshold value used by the presence detection process.
pbImageDataRef	Buffer with the image pixels (greylevel, 8 bits/pixel) of the reference frame.

Return Value

0 → Error.
1 → Ok.

vpmrReadRGB24PD

This function reads the license plate present within an image using presence detection to determine if anything has changed from the previous frame.
The Presence Detection Check can help to save a lot of processing time when there aren't changes in the scene.

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 below).

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 vpmrReadRGB24PD ( long lWidth,
                       long lHeight,
                       unsigned char * pbImageData,
                       bool bFlip,
                       long lMode,
                       long lThreshold,
                       unsigned char* pbImageDataRGBRef );
```

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*).
lMode	This is the operating mode for the Presence Detection. The valid values are: **0** --> Presence detection disabled. **1** --> Presence detection only. **2** --> Presence detection + ROI. **3** --> Presence detection + ROI + Mask.

lThreshold	Sensitivity threshold value used by the presence detection process.
pbImageDataRGBRef	Buffer with the image pixels (RGB, 24 bits/pixel) of the reference frame.

Return Value

0 → Error.
1 → Ok.

vpmrReadRGB32PD

This function reads the license plate present within an image using presence detection to determine if anything has changed from the previous frame.
The Presence Detection Check can help to save a lot of processing time when there aren't changes in the scene.

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 below).

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 vpmrReadRGB32PD ( long lWidth,
                       long lHeight,
                       unsigned char * pbImageData,
                       bool bFlip,
                       long lMode,
                       long lThreshold,
                       unsigned char* pbImageDataRGBRef);
```

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*).
lMode	This is the operating mode for the Presence Detection. The valid values are: **0** --> Presence detection disabled. **1** --> Presence detection only. **2** --> Presence detection + ROI. **3** --> Presence detection + ROI + Mask.

lThreshold	Sensitivity threshold value used by the presence detection process.
pbImageDataRGBRef	Buffer with the image pixels (RGB, 32 bits/pixel) of the reference frame.

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.

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 which result we want to retrieve (starting at 0).

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 which result we want to retrieve (starting at 0).

Return Value

Number of recognized characters.

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 which result we want to retrieve (starting at 0).

Return Value

The Confidence Factor for the last recognition.

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 which result we want to retrieve (starting at 0).

Return Value

Character Confidence Factor.

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 which result we want to retrieve (starting at 0).

Return Value

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

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 which result we want to retrieve (starting at 0).

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	Index of the plate which result we want to retrieve (starting at 0).

vpmrGetFormat

This property returns the country/state code that matches license plate result. If no country is detected it returns 0.

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

Parameters

lPlate Index of the plate which result we want to retrieve (starting at 0).

Return value

Country code of the license plate.
*(See **Appendix 1** for a detailed list of countries codes).*

vpmrGetFormatConfidence

Returns the confidence value for the format (country/state) code detected.

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

Parameters

lPlate Index of the plate which result we want to retrieve (starting at 0).

Return value

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

vpmrGetPolarity

Returns the polarity of the plate.

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

Arguments

lPlate Index of the plate which result we want to retrieve (starting at 0).

Return Value

Polarity of the plate: **1** for dark chars on brighter background, **2** otherwise.

vpmrGetPlateType

Returns the plate type. It returns a numeric code and optionally a text description.

The level of detail and precision of the plate type returned depends on the country/state of issue. Some countries have different plate formats for different types of vehicles and other countries do not.

*(See **Appendix 2** for a detailed list of plate type codes and descriptions).*

```
long vpmrGetPlateType ( char* strDescription,
                        long lDescBufferSize,
                        long lPlate = 0 );
```

Arguments

strDescription	Optional buffer for receiving the plate type description in text (ASCII) format. It this parameter is NULL, no description string is returned.
lDescBufferSize	Size (in bytes) of the buffer passed for receiving the text description of the plate type (if different to NULL).
lPlate	Index of the plate which result we want to retrieve (starting at 0).

Return Value

Plate type numeric code.

vpmrGetAngle

Returns the inclination angle (relative to the horizontal) of the plate.

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

Arguments

lPlate Index of the plate which result we want to retrieve (starting at 0).

Return Value

Angle of inclination (in degrees) of the license plate. 0º means completely horizontal.

vpmrGetNumLines

Returns the number of lines of the plate.

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

Arguments

lPlate Index of the plate which result we want to retrieve (starting at 0).

Return Value

Number of lines contained in the license plate..

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.

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.

Miscellaneous

vpmrGetVersion

Retrieves the VPAR library version numbers.

```
void vpmrGetVersion ( long* plMajor,
                      long* plMinor,
                      long* plRevision,
                      long* plBuild );
```

Arguments

plMajor	Pointer to variable where the major version number will be stored.
plMinor	Pointer to variable where the minor version number will be stored.
plRevision	Pointer to variable where the revision number will be stored.
plBuild	Pointer to variable where the build number will be stored.

Appendix 1 – Country Codes

```
#define LOC_CODE_MULTI  100         // Multi-country.

#define LOC_CODE_ESP    101         // Spain.
#define LOC_CODE_PRT    102         // Portugal.
#define LOC_CODE_FRA    103         // France.
#define LOC_CODE_ITA    104         // Italy.
#define LOC_CODE_GBR    105         // United Kingdom.
#define LOC_CODE_GRC    106         // Greece.
#define LOC_CODE_IRL    107         // Ireland (ROI).
#define LOC_CODE_DEU    108         // Germany.
#define LOC_CODE_AND    109         // Andorra.
#define LOC_CODE_POL    110         // Poland.
#define LOC_CODE_BGR    111         // Bulgaria.
#define LOC_CODE_NLD    112         // Netherlands.
#define LOC_CODE_EST    113         // Estonia.
#define LOC_CODE_BIH    114         // Bosnia-Herzegovina.
#define LOC_CODE_ROU    115         // Romania.
#define LOC_CODE_BEL    116         // Belgium.
#define LOC_CODE_NOR    117         // Norway.
#define LOC_CODE_DNK    118         // Denmark.
#define LOC_CODE_SWE    119         // Sweden.
#define LOC_CODE_FIN    120         // Finland.
#define LOC_CODE_GBZ    121         // Gibraltar.
#define LOC_CODE_CHE    122         // Switzerland.
#define LOC_CODE_AUT    124         // Austria.
#define LOC_CODE_SVN    127         // Slovenia.
#define LOC_CODE_HUN    128         // Hungary.
#define LOC_CODE_CZE    129         // Czech Republic.
#define LOC_CODE_BLR    130         // Belarus.
#define LOC_CODE_UKR    131         // Ukraine.

#define LOC_CODE_CHL    201         // Chile.
#define LOC_CODE_COL    202         // Colombia.
#define LOC_CODE_BRA    203         // Brazil.
#define LOC_CODE_ARG    204         // Argentina.
#define LOC_CODE_MEX    205         // Mexico.
#define LOC_CODE_ECU    206         // Ecuador.
#define LOC_CODE_VEN    207         // Venezuela.
#define LOC_CODE_PER    208         // Peru.
#define LOC_CODE_SLV    209         // El Salvador.
#define LOC_CODE_BOL    210         // Bolivia.
#define LOC_CODE_URY    211         // Uruguay.
#define LOC_CODE_GTM    212         // Guatemala.
#define LOC_CODE_PAN    215         // Panama.
#define LOC_CODE_PRY    216         // Paraguay.
#define LOC_CODE_CRI    217         // Costa Rica.
#define LOC_CODE_DOM    218         // Dominican Republic.
#define LOC_CODE_HND    219         // Honduras.
#define LOC_CODE_NIC    220         // Nicaragua.
#define LOC_CODE_BMU    221         // Bermuda.
#define LOC_CODE_TTO    222         // Trinidad and Tobago.
```

```
#define LOC_CODE_RUS    301         // Russia.
#define LOC_CODE_TUR    302         // Turkey
#define LOC_CODE_VNM    303         // Vietnam.
#define LOC_CODE_IDN    304         // Indonesia.
#define LOC_CODE_PHL    305         // Philippines.
#define LOC_CODE_MYS    306         // Malaysia.
#define LOC_CODE_SGP    307         // Singapore.
#define LOC_CODE_ISR    308         // Israel.
#define LOC_CODE_LBN    309         // Lebanon.
#define LOC_CODE_HKG    310         // Hong Kong.
#define LOC_CODE_MAC    311         // Macau.
#define LOC_CODE_IND    312         // India.
#define LOC_CODE_TWN    313         // Taiwan.
#define LOC_CODE_BHR    314         // Bahrain.
#define LOC_CODE_UAE_AD 315         // Abu Dhabi.
#define LOC_CODE_UAE_DB 316         // Dubai.
#define LOC_CODE_AZE    322         // Azerbaijan.
#define LOC_CODE_KWT    323         // Kuwait.

#define LOC_CODE_ZAF    401         // South Africa.
#define LOC_CODE_MAR    402         // Morocco.
#define LOC_CODE_TUN    403         // Tunisia.
#define LOC_CODE_AGO    404         // Angola.
#define LOC_CODE_SEN    405         // Senegal.
#define LOC_CODE_NGA    406         // Nigeria.
#define LOC_CODE_UGA    407         // Uganda.
#define LOC_CODE_BFA    408         // Burkina Faso.
#define LOC_CODE_TZA    409         // Tanzania.
#define LOC_CODE_KEN    410         // Kenya.

#define LOC_CODE_USA    500         // USA.
#define LOC_CODE_USA_WA 501         // Washington (USA).
#define LOC_CODE_USA_FL 502         // Florida (USA).
#define LOC_CODE_USA_MS 503         // Mississippi (USA).
#define LOC_CODE_USA_NY 504         // New York (USA).
#define LOC_CODE_USA_TX 505         // Texas (USA).
#define LOC_CODE_USA_MA 506         // Massachusetts (USA).
#define LOC_CODE_USA_IL 507         // Illinois (USA).
#define LOC_CODE_USA_CA 508         // California (USA).
#define LOC_CODE_USA_OK 509         // Oklahoma (USA).
#define LOC_CODE_USA_LA 510         // Louisiana (USA).
#define LOC_CODE_USA_NM 511         // New Mexico (USA).
#define LOC_CODE_USA_AR 512         // Arkansas (USA).
#define LOC_CODE_USA_NJ 513         // New Jersey (USA).
#define LOC_CODE_USA_DC 514         // Washington D.C. (USA).
#define LOC_CODE_USA_IN 515         // Indiana (USA).
#define LOC_CODE_USA_ME 516         // Maine (USA).
#define LOC_CODE_USA_NE 517         // Nebraska (USA).
#define LOC_CODE_USA_MO 518         // Missouri (USA).
#define LOC_CODE_USA_NV 519         // Nevada (USA).
#define LOC_CODE_USA_MI 520         // Michigan (USA).
#define LOC_CODE_USA_VA 521         // Virginia (USA).
#define LOC_CODE_USA_NC 522         // North Carolina (USA).
#define LOC_CODE_USA_GA 523         // Georgia (USA).
#define LOC_CODE_USA_PA 524         // Pennsylvania (USA).
```

```
#define LOC_CODE_USA_AK 525         // Alaska (USA).
#define LOC_CODE_USA_MN 526         // Minnesota (USA).
#define LOC_CODE_USA_KY 527         // Kentucky (USA).
#define LOC_CODE_USA_ND 528         // North Dakota (USA).
#define LOC_CODE_USA_OH 529         // Ohio (USA).
#define LOC_CODE_USA_CO 530         // Colorado (USA).
#define LOC_CODE_USA_AZ 531         // Arizona (USA).
#define LOC_CODE_USA_UT 532         // Utah (USA).
#define LOC_CODE_USA_WV 533         // West Virginia (USA).
#define LOC_CODE_USA_CT 534         // Connecticut (USA).
#define LOC_CODE_USA_HI 535         // Hawaii (USA).
#define LOC_CODE_USA_IA 536         // Iowa (USA).
#define LOC_CODE_USA_KS 537         // Kansas (USA).
#define LOC_CODE_USA_SC 538         // South Carolina (USA).
#define LOC_CODE_USA_OR 539         // Oregon (USA).
#define LOC_CODE_USA_MT 540         // Montana (USA).
#define LOC_CODE_USA_AL 541         // Alabama (USA).
#define LOC_CODE_USA_TN 542         // Tennessee (USA).
#define LOC_CODE_USA_RI 543         // Rhode Island (USA).
#define LOC_CODE_USA_NH 544         // New Hampshire (USA).
#define LOC_CODE_USA_DE 545         // Delaware (USA).
#define LOC_CODE_USA_VT 546         // Vermont (USA).
#define LOC_CODE_USA_WI 547         // Wisconsin (USA).
#define LOC_CODE_USA_SD 548         // South Dakota (USA).
#define LOC_CODE_USA_MD 549         // Maryland (USA).
#define LOC_CODE_USA_WY 550         // Wyoming (USA).
#define LOC_CODE_USA_ID 551         // Idaho (USA).

#define LOC_CODE_CAN    561         // Canada.

#define LOC_CODE_AU_ACT 601         // Australian Capital Territory (Australia)
#define LOC_CODE_AU_NSW 602         // New South Wales (Australia)
#define LOC_CODE_AU_QLD 603         // Queensland (Australia)
#define LOC_CODE_AU_SA  604         // South Australia (Australia)
#define LOC_CODE_AU_TAS 605         // Tasmania (Australia)
#define LOC_CODE_AU_VIC 606         // Victoria (Australia)
#define LOC_CODE_AU_WA  607         // Western Australia (Australia)
#define LOC_CODE_AU_NT  608         // Northern Territory (Australia)

#define LOC_CODE_NZL    615         // New Zealand
```

Appendix 2 – Plate Types

```
#define     PLATE_TYPE_UNKNOWN              0     "Unknown"
#define     PLATE_TYPE_GENERIC              1     "Generic"
#define     PLATE_TYPE_MOTORBIKE            2     "Motorbike"
#define     PLATE_TYPE_MOPED                3     "Moped"
#define     PLATE_TYPE_TRUCK                4     "Truck"
#define     PLATE_TYPE_TRANSPORT            5     "Transport"
#define     PLATE_TYPE_HEAVY                6     "Heavy"
#define     PLATE_TYPE_TRAILER              7     "Trailer"
#define     PLATE_TYPE_BUS                  8     "Bus"
#define     PLATE_TYPE_PUBLIC_SERVICE       9     "Public Service"
#define     PLATE_TYPE_TAXI                 10    "Taxi"
#define     PLATE_TYPE_POLICE               11    "Police"
#define     PLATE_TYPE_AMBULANCE            12    "Ambulance"
#define     PLATE_TYPE_FIREFIGHTERS         13    "Firefighters"
#define     PLATE_TYPE_MILITARY             14    "Military"
#define     PLATE_TYPE_DIPLOMATIC           15    "Diplomatic"
#define     PLATE_TYPE_GOVERNMENT           16    "Goverment"
#define     PLATE_TYPE_DEALER               17    "Dealer"
#define     PLATE_TYPE_TEMPORAL             18    "Temporal"
#define     PLATE_TYPE_SPECIAL              19    "Special"
#define     PLATE_TYPE_PERSONALIZED         20    "Personalized"
#define     PLATE_TYPE_AGRICULTURAL         21    "Agricultural"
#define     PLATE_TYPE_DISABLED             22    "Disabled"
#define     PLATE_TYPE_COMMERCIAL           23    "Commercial"
#define     PLATE_TYPE_ASSOCIATION          24    "Association"
```

Appendix 3 - Example of Use

```
void func ()
{
     // Library initialization. We want to read Spanish license plates.
     bool ok = (bool) vpmrInitialize(LOC_CODE_ESP);
     if (ok)
     {
          // Configure character height range (between 25 and 50 pixels).
          vpmrConfigureCharacterHeightRangle(25, 50);

          unsigned char* buffer;  // Image buffer.
          long numchars;          // Number of characters read.
          char text[32];          // String where the result will be stored.
          float fCf;              // Confidence Factor of the reading result.
          float characterCf[32];  // Array to store the characters confidence.

          // Image buffer memory allocation.
          . . .

          // Acquire the image buffer in 256 grayscale levels (1 byte per pixel)
          // and store it into buffer.
          . . .

          // Process image (image dimensions are 384x288 pixels).
          ok = (bool) vpmrRead(1024, 768, buffer);
          if (ok)
          {
               // Retrieve results.
               vpmrGetText(text, 0);
               Cf = vpmrGetGlobalConfidence();
               numchars = vpmrGetNumberOfCharacters();
               for (long i = 0; i < numchars; i++)
               {
                    characterCf[i] = vpmrGetCharacterConfidence(i);
               }
          }
          else
          {
               // Error reading license plate.
          }

          // Read more images.
          . . .

          // Finalize the software
          vpmrEnd();
     }
     else
     {
          // Error initializing the VPAR.
     }
 }
```