

# **VPAR-ESP v.3.0**

## DLL Reference Manual

# Table of Contents

<b>Initialization / Finalization</b>	<b>3</b>
<i>vparespInit</i>	3
<i>vparespEnd</i>	4
<b>Reading Plates</b>	<b>5</b>
<i>vparespRead</i>	5
<i>vparespReadRGB24</i>	6
<i>vparespReadRGB32</i>	7
<i>vparespReadBMP</i>	8
<i>vparespReadJPG</i>	9
<b>Retrieving Results</b>	<b>10</b>
<i>vparespGetNumberOfPlates</i>	10
<i>vparespGetText</i>	11
<i>vparespGetNumberOfCharacters</i>	12
<i>vparespGetGlobalConfidence</i>	13
<i>vparespGetAverageCharacterHeight</i>	14
<i>vparespGetCharacterConfidence</i>	15
<i>vparespGetRectangle</i>	16
<b>Time Management</b>	<b>17</b>
<i>vparespGetProcessingTime</i>	17
<i>vparespSetTimeOut</i>	18
<b>Optional Configuration</b>	<b>19</b>
<i>vparespSetCorrectionCoefficients</i>	19
<i>vparespSetDistortionCorrectionOff</i>	20
<i>vparespConfigureAutomaticCharacterHeight</i>	21
<b>Storing User Data into the HASP Dongle</b>	<b>22</b>
<i>vparespWriteHASP</i>	22
<i>vparespReadHASP</i>	23
<b>Example of Use</b>	<b>24</b>

## Initialization / Finalization

### vparespInit

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 vparespInit ( long lAverageCharacterHeight,  
                  bool bDuplicateLines = false,  
                  bool bSort2LinesPLates = false,  
                  bool bTrailersOn = false );
```

#### Arguments

- 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**.
- bSort2LinesPLates*      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").
- bTrailersOn*      This argument must be true when the library is used to read the two plates present at the back of trailer trucks (the trailer plate and the vehicle plate).

#### Return Value

- 0** → Error.
- 1** → Ok.

## **vparespEnd**

Frees the memory allocated by the **Vehicle Plates Automatic Reader (VPAR)**.

```
void vparespEnd ( void );
```

## Reading Plates

### **vparespRead**

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 greyscale** levels (1 byte per pixel). The *width* and *height* of the image must be supplied as well.

```
long vparespRead ( long IWidth,  
                  long IHeight,  
                  unsigned char * pbImageData );
```

#### Arguments

<i>IWidth</i>	Width (in pixels) of the image that will be analyzed.
<i>IHeight</i>	Height (in pixels) of the image that will be analyzed.
<i>pbImageData</i>	Buffer with the image data (pixels), in 256 grey levels (1 byte per pixel).

#### Return Value

- 0** → Error.
- 1** → Ok.

## vparespReadRGB24

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 vparespReadRGB24 ( long IWidth,  
                        long IHeight,  
                        unsigned char * pbImageData,  
                        bool bFlip = false );
```

### Arguments

<i>IWidth</i>	Width (in pixels) of the image that will be analyzed.
<i>IHeight</i>	Height (in pixels) of the image that will be analyzed.
<i>pbImageData</i>	Buffer with the image data (pixels) using 3 bytes per pixel (RED, GREEN, BLUE).
<i>bFlip</i>	This value must be <b>true</b> 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 ( <i>bottom-up</i> ).

### Return Value

- 0** → Error.
- 1** → Ok.

## vparespReadRGB32

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 vparespReadRGB32 ( long IWidth,  
                        long IHeight,  
                        unsigned char * pbImageData,  
                        bool bFlip = false );
```

### Arguments

<i>IWidth</i>	Width (in pixels) of the image that will be analyzed.
<i>IHeight</i>	Height (in pixels) of the image that will be analyzed.
<i>pbImageData</i>	<i>Buffer</i> with the image data (pixels) using 4 bytes per pixel (RED, GREEN, BLUE, ALPHA).
<i>bFlip</i>	This value must be <b>true</b> 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 ( <i>bottom-up</i> ).

### Return Value

- 0** → Error.
- 1** → Ok.

## vparespReadBMP

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 vparespReadBMP ( char * strFilename );
```

### Arguments

*strFilename*                      Filename of BMP image to process.

### Return Value

**0** → Error.

**1** → Ok.

## **vparespReadJPG**

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 vparespReadJPG ( char * strFilename );
```

### **Arguments**

*strFilename*                                      Filename of JPG image to process.

### **Return Value**

**0** → Error.  
**1** → Ok.

## Retrieving Results

### **vparespGetNumberOfPlates**

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

```
long vparespGetNumberOfPlates ( 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 **vparespInit** 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.

## vparespGetText

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

```
long vparespGetText ( char * strResult,  
                    long IPlate = 0 );
```

### Arguments

*strResult*           String where the ASCII text is returned.

*IPlate*               Index of the plate we want to retrieve.  
If only one plate was read, number **0** must be specified.  
If two plates were 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.

## **vparespGetNumberOfCharacters**

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

```
long vparespGetNumberOfCharacters ( 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 were 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.

## **vparespGetGlobalConfidence**

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 vparespGetGlobalConfidence ( long IPlate = 0 );
```

### **Arguments**

*IPlate*                      Index of the plate we want to retrieve.  
If only one plate was read, number **0** must be specified.  
If two plates were 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.

## **vparespGetAverageCharacterHeight**

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

```
float vparespGetAverageCharacterHeight ( 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 were 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.

## vparespGetCharacterConfidence

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

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

```
float vparespGetCharacterConfidence ( 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 were 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.

## vparespGetRectangle

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

```
void vparespGetRectangle ( long * pLeft,  
                           long * pTop,  
                           long * pRight,  
                           long * pBottom,  
                           long lPlate = 0 );
```

### Arguments

<i>pLeft</i>	X-Coordinate of the Upper-Left corner of the rectangle.
<i>pTop</i>	Y-Coordinate of the Upper-Left corner of the rectangle.
<i>pRight</i>	X-Coordinate of the Lower-Right corner of the rectangle.
<i>pBottom</i>	Y-Coordinate of the Lower-Right corner of the rectangle.
<i>lPlate</i>	Index of the plate we want to retrieve. If only one plate was read, number <b>0</b> must be specified. If two plates were read (a truck with two plates at the back), the result for plate number <b>0</b> or number <b>1</b> can be requested.

# Time Management

## **vparespGetProcessingTime**

Returns the processing time for the last reading operation.

This value is expressed in milliseconds.

```
long vparespGetProcessingTime ( );
```

### **Return Value**

Processing time (in milliseconds) for the last reading.

## **vparespSetTimeOut**

This function specifies the maximum processing time for reading operations.

This value is expressed in milliseconds.

```
void vparespSetTimeOut ( long IMilliseconds );
```

### **Arguments**

*IMilliseconds*                      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 than the time-out specified.

## Optional Configuration

### **vparespSetCorrectionCoefficients**

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 **vparespSetDistortionCorrectionOff** is called.

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

#### **Arguments**

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

## **vparespSetDistortionCorrectionOff**

This function deactivates the distortion correction pre-process.

Use ***vparespSetCorrectionCoefficients*** to activate it again.

```
void vparespSetDistortionCorrectionOff ();
```

## **vparespConfigureAutomaticCharacterHeight**

This function configures the Automatic Character Height steps.

When **vparespInit** is called with the argument *IAvCharacterHeight* 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 VPAR will scan, as well as the order in which they will be scanned.

```
long vparespConfigureAutomaticCharacterHeight ( long INumSteps,  
                                                long * pISteps );
```

### **Arguments**

- |                  |  |
|------------------|--|
| <i>INumSteps</i> | Number of steps that VPAR will perform in automatic height mode (maximum 10 steps).    |
| <i>pISteps</i>   | Array with the heights that VPAR will scan. It must contain <i>INumSteps</i> elements. |

### **Return Value**

- 0** → Error.
- 1** → Ok.

### **Remarks**

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

```
long ISteps[8] = { 25, 30, 35, 40, 45, 50, 55, 60 };  
vparespConfigureAutomaticCharacterHeight (8, ISteps );
```

## Storing User Data into the HASP Dongle

### **vparespWriteHASP**

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 vparespWriteHASP ( unsigned char * pData,  
                        long ISize );
```

#### **Arguments**

*pData* Buffer containing the data to be written in to the HASP internal memory.

*ISize* Size (in bytes) of the data to write (maximum 24 bytes).

#### **Return Value**

**0** → Error.  
**1** → Ok.

## **vparespReadHASP**

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

```
long vparespReadHASP ( 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) vparespInit (30, false, true);
    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 greyscale levels (1 byte per pixel) and store it
        // into buffer.
        ...
        // Image dimensions are 384x288 pixels.

        ok = (bool) vparespRead (384, 288, buffer);
        if (ok)
        {
            vparespGetText (text, 0);
            Cf = vparespGetGlobalConfidence (0);
            numchars = vparespGetNumberOfCharacters (0);
            for (long i = 0; i < numchars; i++)
            {
                characterCf[i] = vparespGetCharacterConfidence (i, 0);
            }
        }
        else
        {
            // Error reading license plate.
        }

        vparespEnd ();
    }
    else
    {
        // Error initializing the VPAR.
    }
}
```