VPARMT

DLL reference manual

Content table

Structures

tRecognitionEngineSettings

This structure is responsible of configure the recognition engine. In every call it configures the engine, so we can change the configuration for multiple configurations.

Parameters

Long milliseconds	Number of maximum milliseconds that the engines process the image. If the value is 0, this parameter does not use it.
Long bApplyCorrection	If this value is 1, the engine applies the following correction parameters.
Float fDistance	Distance between camera and object (meters).
Float fVerticalCoeff	Coefficient to correct the vertical perspective.
Float fHorizontalCoeff	Coefficient to correct the horizontal perspective.
Float fAngle	Angle to correct the rotation (inclination).
Float fRadialCoeff	Coefficient to correct the radial coefficient.
Float fVerticalScrew	Coefficient to correct the vertical screw.
Float fHorizontalScrew	Coefficient to correct the horizontal screw.
Long lNumSteps	Specify if the engine Works with a specific range of character heights. 0 -> not apply, 2 apply the value of firsts positions of vlSteps.
Long vlSteps[8]	Vector with the character heights. Only use the 2 firsts position in case of the value of lNumSteps will be **2**.

The following 4 parameters indicate a ROI inside the image.

Long lLeft, Long lTop	Coordinate of the upper left corner of the area of interest (in pixels).
Long Width, Long Height	Dimensions of ROI (in pixels)
Float fScale	Value of scale for the image. If the value is **1**, it does not apply.

Void lUserParam1*	Pointer to user utilization. This pointer will be returned in the result structure.
Void lUserParam2*	Pointer to user utilization. This pointer will be returned in the result structure.
Void lUserParam3*	Pointer to user utilization. This pointer will be returned in the result structure.
Void lUserParam4*	Pointer to user utilization. This pointer will be returned in the result structure.
Void KillShadow*	Parameter to use the shadow killer function. If the value is 1, this option will be active.
bool CharacterRectangle	Parameter to use the save character rectangle and save image function.

tResults

This structure is returned after every read indicating the result of it.

Parameters

Long lres	Result of the read. **1 ->** correct read, **0** -> wrong read.
Long lNumberOfPlates	Plates read in the image.
Char strResult [MAX_PLATES][MAX_CHAR]	Vector with the read plates.
Long vlNumberOfCharacters [MAX_PLATES]	Vector with the number of character in each plate.
Float vlGlobalConfidence [MAX_PLATES]	Vector with the global confidence of each plate.
Float vfAverageCharacterHeight [MAX_PLATES]	Vector with the average character height of each plate.
Float vlCharacterConfidence [MAX_PLATES][MAX_CHARACTER]	Vector with the character confidence of each plate.

Long vlLeft [MAX_PLATES]	Vector with the left position of each plate.
Long vlTop [MAX_PLATES]	Vector with the top position of each plate.
Long vlRight [MAX_PLATES]	Vector with the right position of each plate.
Long vlBottom [MAX_PLATES]	Vector with the bottom position of each plate.
Long lProcessingTime	Time in milliseconds of the plate detection.
Long vlFormat [MAX_PLATES]	Vector with the format of each plate.
Void lUserParam1*	Pointer returned of the configuration structure.
Void lUserParam2*	Pointer returned of the configuration structure.
Long lPolarity	Polarity of each plate (1 black on white,2 other).Each position represents a plate (units -> plate 0 , tens -> plate 1…)
Float vlFormatConfidence	Format confidence of first plate.
Void EliminateShadow*	Parameter of eliminate shadow function.
Char strPathCorrectedImage [MAX_FILE_PATH]	Path of saved image in case that *CharacterRectangle* function is active.
Long vlCaracterPosition [MAX_PLATES][MAX_CHARACTER][4]	Vector with the character position in case that *CharacterRectangle* function is active.

CandidateRegion

This structure returns the information of the candidate region.

Parameters

Long Left	Left position of candidate region.
Long Top	Top position of candidate region.
Long Right	Right position of candidate region.
Long Bottom	Bottom position of candidate region.
Long ach	Approximate height of the character (pixels).

Initialization / Finalization

vparmtInit

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 vparmtInit(
        long (*callback)(long code, tResults *results),
        long lCountryCode,
        long lAvCharacterHeight ,
        bool bDuplicateLines,
        bool bReserved1,
        long lReserved2,
        bool bTrace);
```

Parameters

Callback	Pointer to callback function. This function will be called asynchronously every time that the engine will have a result.
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**.
lLreserved1	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.

Callback

This function will be called asynchronously every time that the engine will have a result.

long (*callback)(long code, tResults *results)

Parameters

code Code of queued petition. This code is returned in Read function.

results Structure with results.

vparmtEnd

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

void **vparmtEnd** (void);

Vehicle plate reading request

vparmtRead

This function reads the license plate present within an image.

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 vparmtRead (
                    tRecognitionEngineSettings *Configuration,
                    long lWidth,
                    long lHeight,
                    unsigned char * pbImageData );
```

Parameters

Configuration	*Engine configure structure.*
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).

Return value

-1 → Error.
>= **0** → Ok. Code of queued request.

vparmtReadRGB24

This function reads the license plate present within an image.

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 vparmtReadRGB24 (
                        tRecognitionEngineSettings *Configuration,
                        long lWidth,
                        long lHeight,
                        unsigned char * pbImageData,
                        bool bFlip);
```

Parameters

Configuration	*Engine configure structure.*
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).
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

-1 → Error.
>= **0** → Ok. Code of queued request.

vparmtReadRGB32

This function reads the license plate present within an image.

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 vparmtReadRGB32 (
                        tRecognitionEngineSettings *Configuration,
                        long lWidth,
                        long lHeight,
                        unsigned char * pbImageData,
                        bool bFlip );
```

Parameters

Configuration	*Engine configure structure.*
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).
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

-1 → Error.
>= **0** → Ok. Code of queued request.

vparmtReadBMP

This function reads the license plate present within an image.

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

long **vparmtReadBMP** (
tRecognitionEngineSettings *Configuration,
char * *strFilename*);

Parameters

Configuration	*Engine configure structure.*
strFilename	Filename of BMP image to process.

Return value

-1 → Error.
>= **0** → Ok. Code of queued request.

vparmtReadJPG

This function reads the license plate present within an image.

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

```
long vparmtReadJPG (
                     tRecognitionEngineSettings *Configuration,
                     char * strFilename );
```

Parameters

Configuration	*Engine configure structure.*
strFilename	Filename of JPEG image to process.

Return value

-1 → Error.
>= **0** → Ok. Code of queued request.

vparmtReadCam

This function reads the license plate present within an image.

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 vparmtRead (
                    tRecognitionEngineSettings *Configuration,
                    long lWidth,
                    long lHeight,
                    unsigned char * pbImageData,
                    int  idCam );
```

Parameters

Configuration	*Engine configure structure.*
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).
idCam	Camera identifier. If the value is **-1** this function does not use motion detection. If the value is **0 or greater** this function processes the image when motion is detected.

Return value

-1 → Error.
-2 → No motion detected.
>= **0**→ Ok. Code of queued request.

vparmtReadRGB24Cam

This function reads the license plate present within an image.

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 vparmtReadRGB24 (
                        tRecognitionEngineSettings *Configuration,
                        long lWidth,
                        long lHeight,
                        unsigned char * pbImageData,
                        int  idCam,
                        bool bFlip );
```

Parameters

Configuration	*Engine configure structure.*
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).
idCam	Camera identifier. If the value is **-1** this function does not use motion detection. If the value is **0 or greater** this function processes the image when motion is detected.
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

-1 → Error.
-2 → No motion detected.
>= **0** → Ok. Code of queued request.

vparmtReadRGB32Cam

This function reads the license plate present within an image.

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 vparmtReadRGB32 (
                        tRecognitionEngineSettings *Configuration,
                        long lWidth,
                        long lHeight,
                        unsigned char * pbImageData,
                        int  idCam,
                        bool bFlip );
```

Parameters

Configuration	*Engine configure structure.*
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).
idCam	Camera identifier. If the value is **-1** this function does not use motion detection. If the value is **0 or greater** this function processes the image when motion is detected.
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

-1 → Error.
-2 → No motion detected.
>= **0** → Ok. Code of queued request.

vparmtReadBMPCam

This function reads the license plate present within an image.

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

```
long vparmtReadBMP (
                        tRecognitionEngineSettings *Configuration,
                        char * strFilename,
                        int  idCam );
```

Parameters

Configuration	*Engine configure structure.*
strFilename	Filename of BMP image to process.
idCam	Camera identifier. If the value is **-1** this function does not use motion detection. If the value is **0 or greater** this function processes the image when motion is detected.

Return value

-1 → Error.
-2 → No motion detected.
>= **0** → Ok. Code of queued request.

vparmtReadJPGCam

This function reads the license plate present within an image.

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

```
long vparmtReadJPG (
                    tRecognitionEngineSettings *Configuration,
                    char * strFilename,
                    int  idCam );
```

Parameters

Configuration	*Engine configure structure.*
strFilename	Filename of JPEG image to process.
idCam	Camera identifier. If the value is **-1** this function does not use motion detection. If the value is **0 or greater** this function processes the image when motion is detected.

Return value

-1 → Error.
-2 → No motion detected.
>= **0** → Ok. Code of queued request.

vparmtRead_sync

This function reads synchronously the input image and returns the results in a result structure.

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 vparmtRead_sync (
                    tRecognitionEngineSettings *Configuration,
                    long lWidth,
                    long lHeight,
                    unsigned char * pbImageData
                    tResults       *result );
```

Parameters

Configuration	*Engine configure structure.*
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).
Result	Pointer to result structure.

Return value

-1 → Error.
1 → Ok.

vparmtReadRGB24_sync

This function reads synchronously the input image and returns the results in a result structure.

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 vparmtReadRGB24_sync (
                                tRecognitionEngineSettings *Configuration,
                                long lWidth,
                                long lHeight,
                                unsigned char * pbImageData,
                                tResults     *result,
                                bool bFlip );
```

Parameters

Configuration	*Engine configure structure.*
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixels*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixels*).
pbImageData	*Buffer* with the image data (pixels).
Result	Pointer to result structure.
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

-1 → Error.
1 → Ok.

vparmtReadRGB32_sync

This function reads synchronously the input image and returns the results in a result structure.

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 vparmtReadRGB32_sync (
                        tRecognitionEngineSettings *Configuration,
                        long lWidth,
                        long lHeight,
                        unsigned char * pbImageData,
                        tResults  *result,
                        bool bFlip);
```

Parameters

Configuration	*Engine configure structure.*
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixels*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixels*).
pbImageData	*Buffer* with the image data (pixels).
Result	Pointer to result structure.
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

-1 → Error.
1 → Ok.

vparmtReadBMP_sync

This function reads synchronously the input image and returns the results in a result structure.

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

```
long vparmtReadBMP_sync (
                          tRecognitionEngineSettings *Configuration,
                         char * strFilename,
                         tResults       *result);
```

Parameters

Configuration	*Engine configure structure.*
strFilename	Filename of BMP image to process.
Result	Pointer to result structure.

Return value

-1 → Error.
1 → Ok.

vparmtReadJPG_sync

This function reads synchronously the input image and returns the results in a result structure.

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

```
long vparmtReadJPG_sync (
                        tRecognitionEngineSettings *Configuration,
                        char * strFilename,
                        tResults       *result );
```

Parameters

Configuration	*Engine configure structure.*
strFilename	Filename of JPEG image to process.
Result	Pointer to result structure.

Return value

-1 → Error.
1 → Ok.

Motion detection

vparmtSetCameraParameters

Set the motion camera parameters.

```
long vparmtSetCameraParameters (
                                    int id_camera,
                                    long ThresHold,
                                    double Tolerance);
```

Parameters

Id_camera	Camera identifier.
ThresHold	Binarization treshold.
Tolerance	Threshold confidence to accept the motion.

Return value

-1 → Error.
0 → Ok.

vparmtShowCameraMotionWindow

Show a window which the user can view the motion detection.

```
void vparmtShowCameraMotionWindow (
                                    int id_camera,
                                    bool  show);
```

Parameters

Id_camera	Camera identifier.
show	If the value is **TRUE,** a window will be opened.

vparmtDetectMotion8

This function returns the motion detection in a camera.

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.

```
bool vparmtDetectMotion8(
                        long lWidth,
                        long lHeight,
                        unsigned char* pbImageData,
                        int id_camera,
                        int ROILEFT,int ROITOP,int ROIWIDTH,
                        int ROIHEIGHT);
```

Parameters

Id_camera	Camera identifier.
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).
ROILEFT, ROITOP, ROIWIDTH, ROIHEIGHT	Settings parameters of ROI. If the values are **0**, they will not apply.

Return value

True → Motion detected.
False → No motion detected.

vparmtDetectMotion24

This function returns the motion detection in a camera.

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.

```
bool vparmtDetectMotion24(
                                long lWidth,
                                long lHeight,
                                unsigned char* pbImageData,
                                int id_camera,
                                int ROILEFT,int ROITOP,int ROIWIDTH,
                                int ROIHEIGHT);
```

Parameters

Id_camera	Camera identifier.
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).
ROILEFT, ROITOP, ROIWIDTH, ROIHEIGHT	Settings parameters of ROI. If the values are **0**, they will not apply.

Return value

True → Motion detected.
False → No motion detected.

vparmtDetectMotion32

This function returns the motion detection in a camera.

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.

```
bool vparmtDetectMotion32(
                        long lWidth,
                        long lHeight,
                        unsigned char* pbImageData,
                        int id_camera,
                        int ROILEFT,int ROITOP,int ROIWIDTH,
                        int ROIHEIGHT);
```

Parameters

Id_camera	Camera identifier.
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).
ROILEFT, ROITOP, ROIWIDTH, ROIHEIGHT	Settings parameters of ROI. If the values are **0**, they will not apply.

Return value

True → Motion detected.
False → No motion detected.

vparmtDetectMotionFile

This function returns the motion detection in a camera.

```
bool vparmtDetectMotionFile(
                                char* File,
                                int id_camera,
                                int ROILEFT,int ROITOP,int ROIWIDTH,
                                int ROIHEIGHT);
```

Parameters

Id_camera Camera identifier.

File File name.

ROILEFT, ROITOP, ROIWIDTH, ROIHEIGHT Settings parameters of ROI. If the values are **0**, they will not apply.

Return value

True → Motion detected.
False → No motion detected.

Candidate zones

vparmtFindCandidateRegionsFromGreyScale

This function returns the candidate zones to obtain a plate.

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 vparmtFindCandidateRegionsFromGreyscale (
                                    long lWidth,
                                    long lHeight,
                                    unsigned char * pbImageData
                                    long* plNumRegions,
                                    CandidateRegion* pRegions,
                                    long lMaxRegions,
                                    bool zPreciseCoordinates);
```

Parameters

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).
plNumRegions	Pointer to number of candidate region detected.
pRegions	Pointer to candidate regions structures.
lMaxRegions	Number of maximum regions to search.
zPreciseCoordinates	Indicates if the coordinates adjust to plate or this gives a secure margin.

Return value

0 → Error.
1 → Ok.
-**1** → No available cores.

vparmtFindCandidateRegionsFromRGB24

This function returns the candidate zones to obtain a plate.

```
long vparmtFindCandidateRegionsFromRGB24 (
                                    long lWidth,
                                    long lHeight,
                                    unsigned char * pbImageData
                                    long* plNumRegions,
                                    CandidateRegion* pRegions,
                                    long lMaxRegions,
                                    bool zPreciseCoordinates);
```

Parameters

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).
plNumRegions	Pointer to number of candidate region detected.
pRegions	Pointer to candidate regions structures.
lMaxRegions	Number of maximum regions to search.
zPreciseCoordinates	Indicates if the coordinates adjust to plate or this gives a secure margin.

Return value

0 → Error.
1 → Ok.
-**1** → No available cores.

vparmtFindCandidateRegionsFromRGB32

This function returns the candidate zones to obtain a plate.

```
long vparmtFindCandidateRegionsFromRGB32 (
                                    long lWidth,
                                    long lHeight,
                                    unsigned char * pbImageData
                                    long* plNumRegions,
                                    CandidateRegion* pRegions,
                                    long lMaxRegions,
                                    bool zPreciseCoordinates);
```

Parameters

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).
plNumRegions	Pointer to number of candidate region detected.
pRegions	Pointer to candidate regions structures.
lMaxRegions	Number of maximum regions to search.
zPreciseCoordinates	Indicates if the coordinates adjust to plate or this gives a secure margin.

Return value

0 → Error.
1 → Ok.
-**1** → No available cores.

vparmtFindCandidateRegionsFromBMP

This function returns the candidate zones to obtain a plate.

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

```
long vparmtFindCandidateRegionsFromBMP (
                                        long lWidth,
                                        long lHeight,
                                        unsigned char * pbImageData
                                        long* plNumRegions,
                                        CandidateRegion* pRegions,
                                        long lMaxRegions,
                                        bool zPreciseCoordinates);
```

Parameters

file	Path of the file.
plNumRegions	Pointer to number of candidate region detected.
pRegions	Pointer to candidate regions structures.
lMaxRegions	Number of maximum regions to search.
zPreciseCoordinates	Indicates if the coordinates adjust to plate or this gives a secure margin.

Return value

0 → Error.
1 → Ok.
-**1** → No available cores.

vparmtFindCandidateRegionsFromJPG

This function returns the candidate zones to obtain a plate.

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

```
long vparmtReadJPG_sync (
                          char* file,
                          long* plNumRegions,
                          CandidateRegion* pRegions,
                          long lMaxRegions,
                          bool zPreciseCoordinates);
```

Parameters

file	Path of the file.
plNumRegions	Pointer to number of candidate region detected.
pRegions	Pointer to candidate regions structures.
lMaxRegions	Number of maximum regions to search.
zPreciseCoordinates	Indicates if the coordinates adjust to plate or this gives a secure margin.

Return value

0 → Error.
1 → Ok.
-**1** → No available cores.

Miscellaneous

FreeCores

Number of CPU cores available. This is the maximum number of concurrent processes we can manage.

long **FreeCores**();

NumLicenseCores

Number of CPU cores available depending on our license. Maximum parallel processes available for recognition that we can manage.

long **NumLicenseCores**();

vparmtQueueSize

This tells us the number of elements in the queue waiting to be processed by the vehicle plate recognition engine.

long **vparmtQueueSize**();

ActiveLog

This function actives the log.

void **ActiveLog**(bool bActive);

Storing data in HASP dongle

vparmtReadHASP

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

```
long vparmtReadHASP (
                        unsigned char * pData,
                        long lSize );
```

Parameters

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

Return value

0 → Error.
1 → Ok.

vparmtWriteHASP

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 vparmtWriteHASP (
                        unsigned char * pData,
                        long lSize );
```

Parameters

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.