VPARMT

Manual de referencia de la DLL

agosto de 2013

Tabla de Contenidos

Estructuras **4**

tRecognitionEngineSettings 4

tResults 5

CandidateRegion 7

Inicialización / Finalización **8**

vparmtInit 8

Callback 9

vparmtEnd 10

Lectura de Matrículas **11**

vparmtRead 11

vparmtReadRGB24 12

vparmtReadRGB32 13

vparmtReadBMP 14

vparmtReadJPG 15

vparmtReadCam 16

vparmtReadRGB24Cam 17

vparmtReadRGB32Cam 18

vparmtReadBMPCam 19

vparmtReadJPGCam 20

vparmtRead_sync 21

vparmtReadRGB24_sync 22

vparmtReadRGB32_sync 23

vparmtReadBMP_sync 24

vparmtReadJPG_sync 25

Detección de movimiento **26**

vparmtSetCameraParameters 26

vparmtShowCameraMotionWindow 27

vparmtDetectMotion8 28

vparmtDetectMotion24 29

vparmtDetectMotion32 30

vparmtDetectMotionFile 31

Zonas candidatas **32**

vparmtFindCandidateRegionsFromGreyScale 32

vparmtFindCandidateRegionsFromRGB24 33

vparmtFindCandidateRegionsFromRGB32 34

vparmtFindCandidateRegionsFromBMP 35

vparmtFindCandidateRegionsFromJPG 36

Consulta /Miscelánea **37**

FreeCores 37

NumLicenseCores 37

vparmtQueueSize 37

ActiveLog 37

Almacenamiento de datos en el HASP **38**

vparmtWriteHASP 38

vparmtReadHASP 39

Estructuras

tRecognitionEngineSettings

Esta estructura es la encargada de configurar el motor de reconocimiento. Se pasará en cada llamada de lectura de matrículas dándonos la ventaja de poder efectuar diferentes configuraciones en cada llamada.

Parámetros

Long miliseconds	Numero de milisegundos máximo que la librería procesará la imagen. Si el valor es **0** se descartará este parámetro.
Long bApplyCorrection	Si este valor vale **1** aplicará los parámetros de corrección siguientes.
Float fDistance	Distancia aproximada entre la cámara y el objeto (en metros).
Float fVerticalCoeff	Coeficiente para corregir la Perspectiva Vertical.
Float fHorizontalCoeff	Coeficiente para corregir la Perspectiva Horizontal.
Float fAngle	Ángulo para corregir la Rotación (inclinación).
Float fRadialCoeff	Coeficiente para corregir el coeficiente radial.
Float fVerticalScrew	Coeficiente para corregir la rotación Vertical.
Float fHorizontalScrew	Coeficiente para corregir la rotación Horizontal.
Long lNumSteps	Indica si se quiere especificar un rango de altura de caracteres al motor. **0** no aplica, **2** se utilizaran los 2 primeros valores del vector vlSteps.
Long vlSteps[8]	Vector con las alturas de caracteres en pixeles usadas por el motor. Solo se utilizaran las 2 primeras posiciones del vector en caso de que lNumSteps valiera **2**.

Los 4 parametros siguientes indican un ROI dentro de la imagen a procesar.

Long lLeft, Long lTop	Coordenada de la esquina superior izquierda del área de interés (en pixeles).
Long Width, Long Height	Dimensiones del área de interés (en pixeles)

Float fScale	Valor de escalado para la imagen. Este valor se utilizará para ampliar o reducir la imagen dentro del motor. Si el valor es **1** no aplicará.
Void lUserParam1*	Puntero para la utilización del usuario. Este puntero se devolverá en la estructura de resultados.
Void lUserParam2*	Puntero para la utilización del usuario. Este puntero se devolverá en la estructura de resultados.
Void lUserParam3*	Puntero para la utilización del usuario. Este puntero se devolverá en la estructura de resultados.
Void lUserParam4*	Puntero para la utilización del usuario. Este puntero se devolverá en la estructura de resultados.
Void KillShadow*	Parámetro para la utilización de la característica de eliminación de sombras en la matrícula. Si el valor es **1** se utilizará esta opción.
bool CharacterRectangle	Parámetro para la activación de la característica de guardado de rectángulos de cada carácter así como la imagen de la matrícula.

tResults

Esta estructura se devolverá después de una lectura indicando el resultado de esta.

Parámetros

Long lres	Resultado de la lectura. **1 ->** lectura correcta, **0** -> lectura incorrecta.
Long lNumberOfPlates	Numero de matrículas leídas en la imagen.
Char strResult [MAX_PLATES][MAX_CHAR]	Vector con las matrículas leídas.
Long vlNumberOfCharacters [MAX_PLATES]	Vector con el numero de caracteres de cada matrícula.
Float vlGlobalConfidence [MAX_PLATES]	Vector con el % de confianza de cada matrícula.

Float vfAverageCharacterHeight [MAX_PLATES]	Vector con la altura media de carácter de cada matrícula.
Float vlCharacterConfidence [MAX_PLATES][MAX_CHARACTER]	Vector con la altura de cada carácter de las matrículas detectadas.
Long vlLeft [MAX_PLATES]	Vector con la posición izquierda de las matrículas detectadas.
Long vlTop [MAX_PLATES]	Vector con la posición superior de las matrículas detectadas.
Long vlRight [MAX_PLATES]	Vector con la posición derecha de las matrículas detectadas.
Long vlBottom [MAX_PLATES]	Vector con la posición inferior de las matrículas detectadas.
Long lProcessingTime	Tiempo en milisegundos de la detección de matrículas.
Long vlFormat [MAX_PLATES]	Vector con el formato detectado en cada matrícula.
Void lUserParam1*	Puntero devuelto de la estructura de configuración.
Void lUserParam2*	Puntero devuelto de la estructura de configuración.
Void lUserParam3*	Puntero devuelto de la estructura de configuración.
Void lUserParam4*	Puntero devuelto de la estructura de configuración.
Void EliminateShadow*	Parámetro de la utilización de la característica de eliminación de sombras en la matrícula.
Char strPathCorrectedImage [MAX_FILE_PATH]	Ruta donde se guarda la imagen sí hemos activado la opción *CharacterRectangle*.
Long vlCaracterPosition [MAX_PLATES][MAX_CHARACTER][4]	Vector con la posición de cada carácter si hemos activado la opción *CharacterRectangle*.

CandidateRegion

Esta estructura es la encargada de retornar las áreas candidatas a contener una matrícula.

Parámetros

Long Left	Posición izquierda del área candidata.
Long Top	Posición superior del área candidata.
Long Right	Posición derecha del área candidata.
Long Bottom	Posición inferior del área candidata.
Long ach	Altura aproximada del carácter (pixeles).

Inicialización / Finalización

vparmtInit

Inicializa el **Vehicle Plates Automatic Reader** (**VPAR**). Carga las Redes Neuronales Artificiales del OCR e inicializa parámetros. Esta función debe llamarse antes de usar cualquiera de las otras funciones de esta librería.

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

Parámetros

Callback	Puntero a la función de retorno. Esta función se llamará asíncronamente cuando el motor VPAR tenga el resultado listo.
lCountryCode	Código de país para el que queremos reconocer las matrículas. Referirse al fichero de definición para una lista de todos los países soportados.
lAvCharacterHeight	Altura media aproximada de los caracteres de las matrículas a leer (en *pixeles*). Si pasamos un **-1** la librería calcula de manera automática la altura. (de 20 a 50 pixeles) Si se selecciona altura automática, el tiempo de proceso aumenta considerablemente.
bDuplicateLines	Para imágenes capturadas con la mitad de las líneas pasaremos el valor 1 (true). Para los demás casos 0 (false).
lLreserved1	Reordena (cuando el país es España) el resultado de las matrículas de 2 líneas. Por defecto el valor 0 no reordena, con valor 1 se reordena el resultado de la lectura.
lreserved2	Activamos filtro especial para tratamiento de color. Los posibles valores son:

0	Hace la media de los tres canales (**Recomendado, por defecto**)
1	Usa el primer canal (rojo si la imagen/buffer es RGB o azul si es BGR)
2	Usa el segundo canal (en principio, sería siempre el verde)
3	Usa el tercer canal (azul si la imagen/buffer es RGB o rojo si es BGR)
< 0	Error
> 3	Error

bTrace Traza interna. Este parámetro debe valer false.

Valor de Retorno

0 → Error.
1 → Ok.

MUY IMPORTANTE: DEBE RECORDAR QUE SI INICIALIZA LA LIBRERÍA CON EL PARÁMETRO lAvCharacterHeight a -1 , ESTA BUSCARÁ MATRÍCULAS CON CARACTERES ENTRE 20 y 50 PIXELES DE ALTURA.

Callback

Esta función será llamada asíncronamente por el motor VPAR cada vez que tenga un resultado disponible.

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

Parámetros

code Código de identificación de la petición encolada. Este código es retornado al encolar una imagen con las funciones *Read.*

results Estructura con los resultados obtenidos de procesar la imagen con el motor VPAR.

vparmtEnd

Libera la memoria reservada para el **Vehicle Plates Automatic Reader (VPAR)**.

void **vparmtEnd** (void);

Lectura de Matrículas

vparmtRead

Lee la matrícula contenida en una imagen.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixeles* de la imagen en **256 niveles de gris** (1 *byte* por *pixel*).

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen en 256 niveles de grises (1 *byte* por *pixel*).

Valor de Retorno

-1 → Error.
>= **0** → Ok. Codigo de identificación de petición encolada.

vparmtReadRGB24

Lee la matrícula contenida en una imagen.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixeles* de la imagen usando **tres bytes por píxel**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen usando 3 bytes por píxel (Red, Green, Blue).
bFlip	Este valor deberá ser true, sólo si el buffer RGB contiene primero la línea inferior de la imagen y continúa hacia arriba. Algunos dispositivos obtienen el buffer RGB de esta manera (*bottom-up*).

Valor de Retorno

-1 → Error.
>= **0** → Ok. Codigo de identificación de petición encolada.

vparmtReadRGB32

Lee la matrícula contenida en una imagen.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixels* de la imagen usando **cuatro bytes por píxel**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen usando 4 bytes por píxel (Red, Green, Blue, Alfa).
bFlip	Este valor deberá ser true, sólo si el buffer RGB contiene primero la línea inferior de la imagen y continúa hacia arriba. Algunos dispositivos obtienen el buffer RGB de esta manera (*bottom-up*).

Valor de Retorno

-1 → Error.
>= **0** → Ok. Codigo de identificación de petición encolada.

vparmtReadBMP

Lee la matrícula contenida en una imagen.

Como entrada a esta función debe proporcionarse una imagen en **formato BMP**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
strFilename	Nombre del fichero BMP a procesar

Valor de Retorno

-1 → Error.
>= **0** → Ok. Codigo de identificación de petición encolada.

vparmtReadJPG

Lee la matrícula contenida en una imagen.

Como entrada a esta función debe proporcionarse una imagen en **formato JPEG**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
strFilename	Nombre del fichero JPEG a procesar

Valor de Retorno

-1 → Error.
>= **0** → Ok. Codigo de identificación de petición encolada.

vparmtReadCam

Lee la matrícula contenida en una imagen.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixels* de la imagen en **256 niveles de gris** (1 *byte* por *pixel*).

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen en 256 niveles de grises (1 *byte* por *pixel*).
idCam	Identificador de la cámara. Si el valor es **-1** no se usará detección de movimiento. Si el valor es **0 o mayor** se procesará la imagen solo si se ha detectado movimiento para ese identificador de cámara.

Valor de Retorno

-1 → Error.
-2 → No movimiento.
>= **0** → Ok. Codigo de identificación de petición encolada.

vparmtReadRGB24Cam

Lee la matrícula contenida en una imagen.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixels* de la imagen usando **tres bytes por píxel**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen usando 3 bytes por píxel (Red, Green, Blue).
idCam	Identificador de la cámara. Si el valor es **-1** no se usará detección de movimiento. Si el valor es **0 o mayor** se procesará la imagen solo si se ha detectado movimiento para ese identificador de cámara.
bFlip	Este valor deberá ser true, sólo si el buffer RGB contiene primero la línea inferior de la imagen y continúa hacia arriba. Algunos dispositivos obtienen el buffer RGB de esta manera (*bottom-up*).

Valor de Retorno

-1 → Error.
-2 → No movimiento.
>= **0** → Ok. Codigo de identificación de petición encolada.

vparmtReadRGB32Cam

Lee la matrícula contenida en una imagen.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixeles* de la imagen usando **cuatro bytes por píxel**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen usando 4 bytes por píxel (Red, Green, Blue, Alfa).
idCam	Identificador de la cámara. Si el valor es **-1** no se usará detección de movimiento. Si el valor es **0 o mayor** se procesará la imagen solo si se ha detectado movimiento para ese identificador de cámara.
bFlip	Este valor deberá ser true, sólo si el buffer RGB contiene primero la línea inferior de la imagen y continúa hacia arriba. Algunos dispositivos obtienen el buffer RGB de esta manera (*bottom-up*).

Valor de Retorno

-1 → Error.
-2 → No movimiento.
>= **0** → Ok. Codigo de identificación de petición encolada.

vparmtReadBMPCam

Lee la matrícula contenida en una imagen.

Como entrada a esta función debe proporcionarse una imagen en **formato BMP**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
strFilename	Nombre del fichero BMP a procesar
idCam	Identificador de la cámara. Si el valor es **-1** no se usará detección de movimiento. Si el valor es **0 o mayor** se procesará la imagen solo si se ha detectado movimiento para ese identificador de cámara.

Valor de Retorno

-1 → Error.
-2 → No movimiento.
>= **0** → Ok. Codigo de identificación de petición encolada.

vparmtReadJPGCam

Lee la matrícula contenida en una imagen.

Como entrada a esta función debe proporcionarse una imagen en **formato JPEG**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
strFilename	Nombre del fichero JPEG a procesar.
idCam	Identificador de la cámara. Si el valor es **-1** no se usará detección de movimiento. Si el valor es **0 o mayor** se procesará la imagen solo si se ha detectado movimiento para ese identificador de cámara.

Valor de Retorno

-1 → Error.
-2 → No movimiento.
>= **0** → Ok. Codigo de identificación de petición encolada.

vparmtRead_sync

Lee la matrícula contenida en una imagen síncronamente y devuelve los resultados en el puntero a la estructura de resultados.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixeles* de la imagen en **256 niveles de gris** (1 *byte* por *pixel*).

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen en 256 niveles de grises (1 *byte* por *pixel*).
Result	Puntero a la estructura de resultados. Esta estructura será rellenada con los resultados del procesamiento de la imagen.

Valor de Retorno

-1 → Error.
1 → Ok.

vparmtReadRGB24_sync

Lee la matrícula contenida en una imagen síncronamente y devuelve los resultados en el puntero a la estructura de resultados.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixeles* de la imagen usando **tres bytes por píxel**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
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* con los datos (*pixels*) de la imagen usando 3 bytes por píxel (Red, Green, Blue).
Result	Puntero a la estructura de resultados. Esta estructura será rellenada con los resultados del procesamiento de la imagen.
bFlip	Este valor deberá ser true, sólo si el buffer RGB contiene primero la línea inferior de la imagen y continúa hacia arriba. Algunos dispositivos obtienen el buffer RGB de esta manera (*bottom-up*).

Valor de Retorno

-1 → Error.
1 → Ok.

vparmtReadRGB32_sync

Lee la matrícula contenida en una imagen síncronamente y devuelve los resultados en el puntero a la estructura de resultados.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixeles* de la imagen usando **cuatro bytes por píxel**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
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* con los datos (*pixels*) de la imagen usando 4 bytes por píxel (Red, Green, Blue, Alfa).
Result	Puntero a la estructura de resultados. Esta estructura será rellenada con los resultados del procesamiento de la imagen.
bFlip	Este valor deberá ser true, sólo si el buffer RGB contiene primero la línea inferior de la imagen y continúa hacia arriba. Algunos dispositivos obtienen el buffer RGB de esta manera (*bottom-up*).

Valor de Retorno

-1 → Error.
1 → Ok.

vparmtReadBMP_sync

Lee la matrícula contenida en una imagen síncronamente y devuelve los resultados en el puntero a la estructura de resultados.

Como entrada a esta función debe proporcionarse una imagen en **formato BMP**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
strFilename	Nombre del fichero BMP a procesar
Result	Puntero a la estructura de resultados. Esta estructura será rellenada con los resultados del procesamiento de la imagen.

Valor de Retorno

-1 → Error.
1 → Ok.

vparmtReadJPG_sync

Lee la matrícula contenida en una imagen síncronamente y devuelve los resultados en el puntero a la estructura de resultados.

Como entrada a esta función debe proporcionarse una imagen en **formato JPEG**.

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

Parámetros

Configuration	Estructura con la configuración del motor VPAR.
strFilename	Nombre del fichero JPEG a procesar
Result	Puntero a la estructura de resultados. Esta estructura será rellenada con los resultados del procesamiento de la imagen.

Valor de Retorno

-1 → Error.
1 → Ok.

Detección de movimiento

vparmtSetCameraParameters

Ajusta los niveles de calibrado para la detección de movimiento.

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

Parámetros

Id_camera	Id de la cámara a configurar.
TressHold	Umbral de binarización.
Tolerance	Tanto por cierto del umbral de cambio para aceptar la detección de movimiento.

Valor de Retorno

-1 → Error.
0 → Ok.

vparmtShowCameraMotionWindow

Muestra una ventana para el ajuste de los parámetros de detección de movimiento.

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

Parámetros

Id_camera	Id de la cámara a configurar.
show	Si el valor es **TRUE** se mostrará la ventana de ajuste.

vparmtDetectMotion8

Esta función retorna si ha habido un movimiento en la secuencia de imágenes de una cámara.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixeles* de la imagen en **256 niveles de gris** (1 *byte* por *pixel*).

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

Parámetros

Id_camera	Id de la cámara a configurar.
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen en 256 niveles de grises (1 *byte* por *pixel*).
ROILEFT, ROITOP, ROIWIDTH, ROIHEIGHT	Parámetros de definición del área de interés a aplicar. Si estos valores valen **0** no se aplicará el área de interés.

Valor de Retorno

True → Ha habido movimiento.
False → No ha habido movimiento.

vparmtDetectMotion24

Esta función retorna si ha habido un movimiento en la secuencia de imágenes de una cámara.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixeles* de la imagen usando **tres bytes por píxel**.

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

Parámetros

Id_camera	Id de la cámara a configurar.
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen usando 3 bytes por píxel (Red, Green, Blue).
ROILEFT, ROITOP, ROIWIDTH, ROIHEIGHT	Parámetros de definición del área de interés a aplicar. Si estos valores valen **0** no se aplicará el área de interés.

Valor de Retorno

True → Ha habido movimiento.
False → No ha habido movimiento.

vparmtDetectMotion32

Esta función retorna si ha habido un movimiento en la secuencia de imágenes de una cámara.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixeles* de la imagen usando **4 bytes por píxel**.

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

Parámetros

Id_camera	Id de la cámara a configurar.
lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen usando 4 bytes por píxel (Red, Green, Blue, Alfa).
ROILEFT, ROITOP, ROIWIDTH, ROIHEIGHT	Parámetros de definición del área de interés a aplicar. Si estos valores valen **0** no se aplicará el área de interés.

Valor de Retorno

True → Ha habido movimiento.
False → No ha habido movimiento.

vparmtDetectMotionFile

Esta función retorna si ha habido un movimiento en la secuencia de imágenes de una cámara.

```
bool vparmtDetectMotion8(
                                char* File,
                                int id_camera,
                                int ROILEFT,int ROITOP,int ROIWIDTH,
                                int ROIHEIGHT);
```

Parámetros

Id_camera	Id de la cámara a configurar.
File	Nombre del fichero.
ROILEFT, ROITOP, ROIWIDTH, ROIHEIGHT	Parámetros de definición del área de interés a aplicar. Si estos valores valen **0** no se aplicará el área de interés.

Valor de Retorno

True → Ha habido movimiento.
False → No ha habido movimiento.

Zonas candidatas

vparmtFindCandidateRegionsFromGreyScale

Devuelve las zonas candidatas a obtener una matrícula.

Como entrada a esta función debe proporcionarse el *buffer* de la imagen a analizar. Este *buffer* corresponde a los *pixeles* de la imagen en **256 niveles de gris** (1 *byte* por *pixel*).

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

Parámetros

lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen en 256 niveles de grises (1 *byte* por *pixel*).
plNumRegions	Puntero donde se devolverá el numero de regiones candidatas encontradas.
pRegions	Puntero a la estructura con las regiones candidatas encontradas.
lMaxRegions	Numero de regiones máximas a buscar.
zPreciseCoordinates	Si este valor es **TRUE** las coordenadas del ajuste de matrícula se ceñirán a esta. Si es **FALSE** se dispondrá de un margen de seguridad en la zona de detección.

Valor de Retorno

0 → Error.
1 → Ok.
-**1** → No hay núcleos disponibles.

vparmtFindCandidateRegionsFromRGB24

Devuelve las zonas candidatas a obtener una matrícula.

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

Parámetros

lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen.
plNumRegions	Puntero donde se devolverá el numero de regiones candidatas encontradas.
pRegions	Puntero a la estructura con las regiones candidatas encontradas.
lMaxRegions	Numero de regiones máximas a buscar.
zPreciseCoordinates	Si este valor es **TRUE** las coordenadas del ajuste de matrícula se ceñirán a esta. Si es **FALSE** se dispondrá de un margen de seguridad en la zona de detección.

Valor de Retorno

0 → Error.
1 → Ok.
-**1** → No hay núcleos disponibles.

vparmtFindCandidateRegionsFromRGB32

Devuelve las zonas candidatas a obtener una matrícula.

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

Parámetros

lWidth	Anchura de la imagen que será analizada por el VPAR (en p*ixeles*).
lHeight	Altura de la imagen que será analizada por el VPAR (en p*ixeles*).
pbImageData	*Buffer* con los datos (*pixeles*) de la imagen.
plNumRegions	Puntero donde se devolverá el numero de regiones candidatas encontradas.
pRegions	Puntero a la estructura con las regiones candidatas encontradas.
lMaxRegions	Numero de regiones máximas a buscar.
zPreciseCoordinates	Si este valor es **TRUE** las coordenadas del ajuste de matrícula se ceñirán a esta. Si es **FALSE** se dispondrá de un margen de seguridad en la zona de detección.

Valor de Retorno

0 → Error.
1 → Ok.
-**1** → No hay núcleos disponibles.

vparmtFindCandidateRegionsFromBMP

Devuelve las zonas candidatas a obtener una matrícula.

Como entrada a esta función debe proporcionarse una imagen en **formato BMP**.

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

Parámetros

file	Path del archivo.
plNumRegions	Puntero donde se devolverá el numero de regiones candidatas encontradas.
pRegions	Puntero a la estructura con las regiones candidatas encontradas.
lMaxRegions	Numero de regiones máximas a buscar.
zPreciseCoordinates	Si este valor es **TRUE** las coordenadas del ajuste de matrícula se ceñirán a esta. Si es **FALSE** se dispondrá de un margen de seguridad en la zona de detección.

Valor de Retorno

0 → Error.
1 → Ok.
-**1** → No hay núcleos disponibles.

vparmtFindCandidateRegionsFromJPG

Devuelve las zonas candidatas a obtener una matrícula.

Como entrada a esta función debe proporcionarse una imagen en **formato JPEG**.

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

Parámetros

file	Path del archivo
plNumRegions	Puntero donde se devolverá el numero de regiones candidatas encontradas.
pRegions	Puntero a la estructura con las regiones candidatas encontradas.
lMaxRegions	Numero de regiones máximas a buscar.
zPreciseCoordinates	Si este valor es **TRUE** las coordenadas del ajuste de matrícula se ceñirán a esta. Si es **FALSE** se dispondrá de un margen de seguridad en la zona de detección.

Valor de Retorno

0 → Error.
1 → Ok.
-**1** → No hay núcleos disponibles.

Consulta /Miscelánea

FreeCores

Esta función nos devuelve el número de núcleos libres utilizables por VPAR.

long **FreeCores**();

NumLicenseCores

Esta función nos devuelve el número de licenciados del motor VPAR.

long **NumLicenseCores**();

vparmtQueueSize

Esta función nos el número de elementos encolados pendientes de procesar por el motor VPAR.

long **vparmtQueueSize**();

ActiveLog

Esta función activa los logs de la librería multithread. Se recomiendo no activar los logs.

void **ActiveLog**(bool bActive);

Almacenamiento de datos en el HASP

vparmtWriteHASP

Escribe datos en la memoria interna de la mochila de protección (dongle). Esta capacidad de almacenar datos en el HASP puede ser usada por el usuario para cualquier propósito. Los datos son encriptados automáticamente antes de escribirse en la memoria del HASP.

Pueden almacenarse un máximo de 24 bytes.

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

Parámetros

pData	Buffer con los datos a escribir en la memoria del HASP.
lSize	Tamaño (en bytes) de los datos a escribir (máximo 24).

Valor de Retorno

0 → Error.
1 → Ok.

vparmtReadHASP

Lee los datos almacenados en la memoria interna de la mochila de protección (dongle). Esta capacidad de almacenar datos en el HASP puede ser usada por el usuario para cualquier propósito. Los datos son desencriptados automáticamente después de leerse de la memoria del HASP.

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

Parámetros

pData	Buffer donde se almacenarán los datos leídos de la memoria del HASP.
lSize	Tamaño (en bytes) de los datos a leer.

Valor de Retorno

0 → Error.
1 → Ok.