CIDAR v3

DLL Reference Manual

September 18

Table of Contents

Initialization / Finalization

cidarInit

```
long cidarInitialize( long lAvCharacterHeight = -1,
                      bool zDuplicateLines = false,
                      bool zTrace = false );
```

Purpose: Initalizes the Container ID Automatic Reader. Allocates Memory, loads the Artificial Neural Networks, etc...

Parameters:

lAvCharacterHeight	Aproximate average height of the characteres within the images (in pixels). **-1** for Automatic (finds charaters between **25** and **100** pixels in height).
bDuplicateLines	This parameter must be **true** when the imaging hardware only acquires half of the horizontal lines. **false** otherwise.
zTrace	This parameter must be set to **false**.

Return Value:

OK: 1
ERROR: 0

cidarInitialize

```
long cidarInitialize( bool zTrace = false );
```

This is a simplified version of ***cidarInit***.

Purpose: Initalizes the Container ID Automatic Reader. Allocates Memory, loads the Artificial Neural Networks, etc...

Parameters:

zTrace	This parameter must be set to false.

Return Value:

OK: 1
ERROR: 0

cidarEnd

```
void cidarEnd( void );
```

Purpose: Frees memory and closes the CIDAR.

Return Value:

OK: 1
ERROR: 0

Library Configuration

cidarSetCorrectionCoefficients

```
void cidarSetCorrectionCoefficients(  float fDistance,
                                      float fVerticalCoeff,
                                      float fHorizontalCoeff,
                                      float fRadialCoeff,
                                      float fAngle,
                                      float fVerticalSkew = 0.0,
                                      float fHorizontalSkew = 0.0 );
```

Purpose: 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 andor vertical perspective); radial distortion (fisheye effect), horizontal and vertical skew and rotation.

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

Parameters:

fDistance	Approximate distance between camera and object (in meters).
fVerticalCoeff	Coefficient for correcting Vertical Perspective.
fHorizontalCoeff	Coefficient for correcting Horizontal Perspective.
fRadialCoeff	Coefficient for correcting Fisheye effect.
fAngle	Angle for correcting Rotation.
fVerticalSkew	Ammount of Vertical Skew.
fHorizontalSkew	Ammount of Horizontal Skew.

cidarSetDistortionCorrectionOff

```
void cidarSetDistortionCorrectionOff( );
```

Purpose: Deactivates the distortion correction pre-process.

Use ***cidarSetCorrectionCoefficients*** to activate it again.

cidarConfigureAutomaticCharacterHeight

```
long cidarConfigureAutomaticCharacterHeight( long lMinCharHeight,
                                             long lMaxCharHeight );
```

Purpose: Configures the Automatic Character Height range.

When cidarInit is called with the argument lAvCharacterHeight set to **-1** the automatic character height mode is selected.

By default, the range of character heights scanned in this mode is from 25 pixels to 100 pixels.

By using this function, the user can select a different height range.

Parameters:

lMinCharHeight: Minimum character height (in pixels) of the characters to read.

lMaxCharHeight: Maximum character height (in pixels) of the characters to read.

Return Value:

OK: 1
ERROR: 0

cidarSetScaleFactor

```
long cidarSetScaleFactor( float fScale );
```

Purpose: Scales the image internally before procesing it.

Parameters:

fScale Scale factor. Width and height dimensions are multiplied by this factor.

Return Value:

OK: 1
ERROR: 0

cidarSetRectangle

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

Purpose: Sets the rectangle where the inspection process will take place.

Only the provided region will be inspected.

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

Parameters: ***lLeft, lTop*** Top and left coordinates of the rectangle (in pixels).

lWidth, lHeight Rectangle dimensions (in pixels).

Return Value:

OK: 1
ERROR: 0

Remarks: The inspection rectangle specified will be applied to all the further inspections until cidarSetRectangle is called again with different parameters.

In order to inspect the whole image, cidarSetRectangle must be called with all its parameters set to 0: **cidarSetRectangle(0, 0, 0, 0);**

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

cidarConfigureCodeFormat

```
long cidarConfigureCodeFormat(   long lOrientation,
                                 long lLines,
                                 long lPolarity  );
```

Purpose: Informs the CIDAR library about the format of the ID codes that will be processed. It is highly desirable to limit the scope of the codes to recognize, in order to reduce the procesing time and increase the effciency of the engine.

Parameters:

lOrientation This parameter informs the library about the possible orientation of the codes to be recognized. Valid values:

CODE_FORMAT_HORIZONTAL (1) : Only horizontal codes.
CODE_FORMAT_VERTICAL (2) : Only vertical codes.
CODE_FORMAT_HORIZONTAL_OR_VERTICAL (3) : Horizontal and vertical codes.

lLines This parameter informs the library about the possible number of lines for the codes. Valid values:

CODE_FORMAT_ONE_LINE (1) : Only one-line codes.
CODE_FORMAT_TWO_LINES (2) : Only two-lines codes.
CODE_FORMAT_FOUR_LINES (4) : Only four-lines codes.
CODE_FORMAT_ANY_LINES (-1) : Any number of lines.

lPolarity This parameter informs the library about the possible polarity of the characters over the background. Valid values:

CODE_FORMAT_BLACK_ON_WHITE (1) : Characters darker than background.
CODE_FORMAT_WHITE_ON_BLACK (2) : Characters lighter than background.
CODE_FORMAT_ANY_ON_ANY (3) : Characters darker or lighter than background.
CODE_FORMAT_BIPOLAR (4) : CODE_FORMAT_ANY_ON_ANY plus bioplar codes.

Return Value:

OK: 1
ERROR: 0

SlantCorrectionOn

```
void cidarSlantCorrectionOn( );
```

Purpose: Activates the automatic slant detection and correction.

SlantCorrectionOff

```
void cidarSlantCorrectionOff( );
```

Purpose: Deactivates the automatic slant detection and correction.

cidarReturnSpacesOn

```
void cidarReturnSpacesOn( );
```

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

cidarReturnSpacesOff

```
void cidarReturnSpacesOff( );
```

Purpose: By calling this function after Init the number plates recognized will be returned without the spaces between characters. This is the default setting.

cidarConfigurePreprocessing

```
void cidarConfigurePreprocessing( long lOperation );
```

Purpose: Configures the image preprocessing for CIDAR.

Parameters:

lOperation Preprocessing operation. Valid values for this parameter are:

1	Histogram equalization.
2	Histogram dynamic range strecth.
3	All pixels multiplied by 2.
4	Median filter.
5	Horizontal low pass filter.
6	Vertical low pass filter.
7	Low pass filter.

cidarConfigureSearchAngle

```
void cidarConfigureSearchAngle( float fAngle );
```

Purpose: Configures the angle range for locating the code.

Parameters:

fAngle Search angle (absolute value).

cidarProcessExtraInfo

```
void cidarProcessExtraInfo( bool zEnable );
```

Purpose: This function enablesdisables the search and recognition of the extra characters included in the container code.

Parameters:

zEnable If true, the Extra Info procesing will be enabled.

Reading Container ID Codes

cidarRead

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

Purpose: Analyzes an input image, finds the Container ID code and recognizes the characters.

Parameters:

lWidth	Image Width (in pixels).
lHeight	Image Height (in pixels).
pbImageData	Buffer with the image pixels (greylevel, 8 bitspixel).

Return Value:

OK: 1
ERROR: 0

cidarReadPD

```
long cidarReadPD ( long lWidth,
                   long lHeight,
                   unsigned char* pbImageData,
                   long lMode,
                   long lThreshold,
                   unsigned char* pbImageDataRef = 0 );
```

Purpose: Analyzes an input image, finds the Container ID code and recognizes the characters using presence detection.

Parameters:

lWidth Image Width (in pixels).

lHeight Image Height (in pixels).

pbImageData Buffer with the image pixels (greylevel, 8 bitspixel).

lMode 0 --> Presence detection disabled.
1 --> Presence detection only.
2 --> Presence detection + ROI.
3 --> Presence detection + ROI + Mask.

lThreshold Threshold value used by the presence detection process.

pbImageDataRef Buffer with the image pixels (greylevel, 8 bits/pixel) of the reference frame.

Return Value:

OK: 1
ERROR: 0

cidarReadRGB24

```
long cidarReadRGB24(  long lWidth,
                      long lHeight,
                      unsigned char* pbImageDataRGB,
                      bool zFlip = false );
```

Purpose: Analyzes an input image from a 24 bits RGB buffer, finds the Container ID code and recognizes the characters characters.

Parameters:

lWidth	Image Width (in pixels).
lHeight	Image Height (in pixels).
pbImageDataRGB	Buffer with the image pixels (RGB, 24 bits pixel).
zFlip	true to perform a Vertical Flip of the image buffer.

Return Value:

OK: 1
ERROR: 0

cidarReadRGB24PD

```
long cidarReadRGB24PD( long lWidth,
                       long lHeight,
                       unsigned char* pbImageDataRGB,
                       bool zFlip,
                       long lMode,
                       long lThreshold,
                       unsigned char* pbImageDataRGBRef = 0 );
```

Purpose: Analyzes an input image from a 24 bits RGB buffer, finds the Container ID code and recognizes the characters using presence detection.

Parameters:

lWidth	Image Width (in pixels).
lHeight	Image Height (in pixels).
pbImageDataRGB	Buffer with the image pixels (RGB, 24 bits pixel).
zFlip	true to perform a Vertical Flip of the image buffer.
lMode	0 --> Presence detection disabled. 1 --> Presence detection only. 2 --> Presence detection + ROI. 3 --> Presence detection + ROI + Mask.
lThreshold	Threshold value used by the presence detection process.
pbImageDataRGBRef	Buffer with the image pixels (RGB, 24 bits pixel) of the reference frame.

Return Value:

OK: 1
ERROR: 0

cidarReadRGB32

```
long cidarReadRGB32(  long lWidth,
                      long lHeight,
                      unsigned char* pbImageDataRGB,
                      bool zFlip = false );
```

Purpose: Analyzes an input image from a 32 bits RGB buffer, finds the Container ID code and recognizes the characters.

Parameters:

lWidth	Image Width (in pixels).
lHeight	Image Height (in pixels).
pbImageDataRGB	Buffer with the image pixels (RGB, 32 bits pixel).
zFlip	true to perform a Vertical Flip of the image buffer.

Return Value:

OK: 1
ERROR: 0

cidarReadRGB32PD

```
long cidarReadRGB32PD ( long lWidth,
                        long lHeight,
                        unsigned char* pbImageDataRGB,
                        bool zFlip,
                        long lMode,
                        long lThreshold,
                        unsigned char* pbImageDataRGBRef = 0 );
```

Purpose: Analyzes an input image from a 32 bits RGB buffer, finds the Container ID code and recognizes the characters using presence detection.

Parameters:

lWidth	Image Width (in pixels).
lHeight	Image Height (in pixels).
pbImageDataRGB	Buffer with the image pixels (RGB, 32 bits pixel).
zFlip	true to perform a Vertical Flip of the image buffer.
lMode	0 --> Presence detection disabled. 1 --> Presence detection only. 2 --> Presence detection + ROI. 3 --> Presence detection + ROI + Mask.
lThreshold	Threshold value used by the presence detection process.
pbImageDataRGBRef	Buffer with the image pixels (RGB, 32 bits pixel) of the reference frame.

Return Value:

OK: 1
ERROR: 0

cidarReadBMP

```
long cidarReadBMP( char* pcFilename );
```

Purpose: Analyzes an input image from a BMP file, finds the Container ID code and recognizes the characters.

Parameters:

pcFilename Filename of a standard BMP file (24 bitspixel).

Return Value:

OK: 1
ERROR: 0

cidarReadJPG

```
long cidarReadJPG( char* pcFilename );
```

Purpose: Analyzes an input image from a JPG file, finds the Container ID code and recognizes the characters.

Parameters:

pcFilename Filename of a standard JPG file.

Return Value:

OK: 1
ERROR: 0

cidarReadJPGBuffer

```
long cidarReadJPGBuffer( char* pbImageData,
                         long lSize );
```

Purpose: Analyzes an input image from a JPG-encoded buffer, finds the Container ID code and recognizes the characters.

Parameters:

pbImageData	JPG encoded buffer.
lSize	Buffer size (in bytes)

Return Value:

OK: 1
ERROR: 0

Retrieving Results

cidarCodeFound

```
long cidarGetNumCodes( );
```

Purpose: Returns number of codes found.

Return Value: Number of ID codes found

cidarCodeFound

```
bool cidarCodeFound( );
```

Purpose: Returns whether a code was found or not.

Return Value:
Code found : true
Code not found : false

cidarCodeVerified

```
bool cidarCodeVerified( long lCodeIndex = 0 );
```

Purpose: Returns whether the code was verified successfully or not.

Parameters:
lCodeIndex 0-based index of the code.

Return Value:
Code Checksum verified : true
Code Checksum not verified : false

cidarGetCode

```
long cidarGetCode( char* strResult,
                   long lCodeIndex = 0 );
```

Purpose: Returns the container ID code read.

Parameters:

strResult Text string with the result.

lCodeIndex 0-based index of the code.

Return Value:

OK: 1
ERROR: 0

cidarGetNumberOfCharacters

```
long cidarGetNumberOfCharacters( long lCodeIndex = 0 );
```

Purpose: Returns the number of characters in the last Container ID code read.

Parameters:

lCodeIndex 0-based index of the code.

Return Value: Number of Characters.

cidarGetGlobalConfidence

```
float cidarGetGlobalConfidence( long lCodeIndex = 0 );
```

Purpose: Returns the confidence factor (0 - 100%) for the last Container ID code read.

Parameters:

lCodeIndex 0-based index of the code.

Return Value: Overall Confidence Factor.

cidarGetCharacterConfidence

```
float cidarGetCharacterConfidence( long lCharIndex,
                                   long lCodeIndex = 0 );
```

Purpose: Returns the confidence factor (0 - 100%) for a specific character.

Parameters:

lCharIndex Index of the character [0 .. NumberOfCharacters-1]

lCodeIndex 0-based index of the code.

Return Value: Character Confidence Factor.

cidarGetAverageCharacterHeight

```
float cidarGetAverageCharacterHeight( long lCodeIndex = 0 );
```

Purpose: Returns the average height in pixels of the characters recognized in the last Container ID code read.

Parameters:

lCodeIndex 0-based index of the code.

Return Value: Average Character Height.

cidarGetRectangle

```
void cidarGetRectangle( long* plLeft,
                        long* plTop,
                        long* plRight,
                        long* plBottom,
                        long lCodeIndex = 0 );
```

Purpose: Returns the coordinates of the rectangle containing the last Container ID code read.

Parameters:

plLeft	The LEFT coordinate (or X1) is returned here.
plTop	The TOP coordinate (or Y1) is returned here.
plRight	The RIGHT coordinate (or X2) is returned here.
plBottom	The TOP coordinate (or Y2) is returned here.
lCodeIndex	0-based index of the code.

cidarGetOrientation

```
long cidarGetOrientation( long lCodeIndex = 0 );
```

Purpose: Returns the orienation of the code.

Parameters:

lCodeIndex	0-based index of the code.

Return Value:

1 = Horizontal.
2 = Vertical.

cidarGetPolarity

```
long cidarGetPolarity( long lCodeIndex = 0 );
```

Purpose: Returns the polarity of the last code found.

Parameters:

lCodeIndex	0-based index of the code.

Return Value:

1 = Dark characters on brighter background.
2 = Bright characters on darker background.

cidarGetNumLines

```
long cidarGetNumLines( long lCodeIndex = 0 );
```

Purpose: Returns the number of lines of the code.

The number of lines can be 1 or 2.

Parameters:

lCodeIndex 0-based index of the code.

Return Value: Number of lines for the code found (1 or 2 lines).

cidarExtraInfoFound

```
bool cidarExtraInfoFound( long lCodeIndex = 0 );
```

Purpose: Returns whether the extra info characters were found or not.

Parameters:

lCodeIndex 0-based index of the code.

Return Value:

Extra info found : true
Extra info not found : false

cidarGetExtraInfo

```
long cidarGetExtraInfo( char* strResult,
                        long lCodeIndex = 0 );
```

Purpose: Returns the container ID extra info read.

Parameters:

strResult Text string with the result.

lCodeIndex 0-based index of the code.

Return Value:
OK: 1
ERROR: 0

cidarGetExtraInfoNumberOfCharacters

```
long cidarGetExtraInfoNumberOfCharacters( long lCodeIndex = 0 );
```

Purpose: Returns the number of characters in the last Container ID extra info read.

Parameters:
lCodeIndex 0-based index of the code.

Return Value: Number of Characters.

cidarGetExtraInfoConfidence

```
float cidarGetExtraInfoConfidence( long lCodeIndex = 0 );
```

Purpose: Returns the confidence factor (0 - 100%) for the last Container ID extra info read.

Parameters:
lCodeIndex 0-based index of the code.

Return Value: Extra info Confidence Factor.

cidarGetExtraInfoCharacterConfidence

```
float cidarGetExtraInfoCharacterConfidence( long lCharIndex,
                                            long lCodeIndex = 0 );
```

Purpose: Returns the confidence factor (0 - 100%) for a specific character of the Extra Info code.

Parameters:

lCharIndex Index of the character [0 .. NumberOfExtraInfoCharacters-1]

lCodeIndex 0-based index of the code.

Return Value: Extra Info character Confidence Factor.

cidarGetExtraInfoRectangle

```
void cidarGetExtraInfoRectangle( long* plLeft,
                                 long* plTop,
                                 long* plRight,
                                 long* plBottom,
                                 long lCodeIndex = 0 );
```

Purpose: Returns the coordinates of the rectangle containing the last Container ID extra info read.

Parameters:

plLeft The LEFT coordinate (or X1) is returned here.

plTop The TOP coordinate (or Y1) is returned here.

plRight The RIGHT coordinate (or X2) is returned here.

plBottom The TOP coordinate (or Y2) is returned here.

lCodeIndex 0-based index of the code.

Time Management

cidarGetProcessingTime

```
long cidarGetProcessingTime( );
```

Purpose: Returns the time taken to process the last image.

Return Value: Process time in milliseconds.

cidarSetTimeOut

```
void cidarSetTimeOut( unsigned long dwMilliseconds );
```

Purpose: Sets a maximum processing time for the "Read" functions.

If the process has not finished after the specified timeout, the "Read" functions will time-out and quit.

Parameters:

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

cidarWriteHASP

```
long cidarWriteHASP(BYTE* pData, long lSize);
```

Purpose: 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.

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:

OK: 1
ERROR: 0

cidarReadHASP

```
long cidarReadHASP(BYTE* pData, long lSize);
```

Purpose: Reads the data stored in the internal memory of the HASP dongle.

Data is automatically decrypted after being read from the HASP memory.

Parameters:

pData Buffer where the retrieved data will be stored.

lSize Size (in bytes) of the data to read.

Return Value:

OK: 1
ERROR: 0

Miscellaneous Functions

cidarGetVersion

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

Purpose: Retrieves the version numbers.

Parameters:

plMajor	Pointer where the major version number is returned.
plMinor	Pointer where the minor version number is returned.
plRevision	Pointer where the version revision number is returned.
plBuild	Pointer where the version build number is returned.