ChequeCOM

Check SDK Library Description

July 2013

Table of Contents

Introduction

The ChequeCOM library is an SDK that gives your application an easy way to extract data from bank checks. Using the library, the integrator can detect a paper insertion in the scanner (ScanShell® 800/800N/1000), scan the check, process the image and retrieve the different data fields that are printed on the check.

ChequeCOM may work as a stand-alone SDK or can be integrated with the ScanW SDK, thus giving a complete solution to scan checks and other documents.

Retrieving Information from Checks

Retrieving the data from a check involves 3 steps:

- Loading the internal image by scanning the check.
- Processing the internal image – Use ProcessCheque to activate the check analyzer on the scanner's internal image.
- Retrieve the data – Use the GetChequeData function to retrieve the detected data.

IMPORTANT:
The internal image is overwritten in every new scan.

Functions and Properties Summary

Name	Type	Functionality
PaperInTray	Property	Checks if a document is in the scanner tray
PressedButton	Property	Returns which of the 3 scanner's button is pressed (for ScanShell® 1000 only)
ScannerType	Property	Returns the connected scanner type (ScanShell® 800/800N/1000)
IsNeedCalibration	Property	Checks if the scanner needs to be calibrated
IsScannerValid	Property	Verifies the scanner functionality
JpgQuality	Property	Sets the JPEG compression factor
ScanProg	Property	Returns the scan progress (when using ScanToFileEx)
InitChequeLibrary	Function	Activates and enables the library functionality
CalibrateScanner	Function	Calibrates the scanner color sensor
Clean	Function	Cleans the scanner
ScanToFile	Function	Scans document to a bitmap file
ScanToFileEx	Function	Scans document while showing the progress bar
ProcessCheque	Function	Processes the recently scanned check
GetChequeData	Function	Returns a specific data item value
GetRawDataEx	Function	Returns the raw text data from the document
GetBwImage	Function	Saves the scanned image in black and white format
ConvertImage	Function	Image file conversion function
ReformatImage	Function	Image file conversion function
RotateImage	Function	Image file conversion function

Licensing

The use of the **ChequeCOM** library *must* be under two conditions:

- Running while the scanner (ScanShell® 800/800N/1000) is connected to the PC as the scanner also functions as a hardware plug.
- Initializing the library with the license before making any other library function calls. See more details in the section that describes the function *Init**Cheque**Library().*

Distribution

To install the SDK files at the destination computer, you simply need to copy all the SDK files that are in the SDK installation folder to the destination computer.

There are some files that will need to be registered on the destination computer such as COM\ActiveX objects. Install these files at the end of the SDK files installation since it will need the none COM\ActiveX files to exist before registration.

Here is a list of the files that need to be registered:

- CheckCom.dll
- ScanW.dll (Com object)
- ScanWEx.dll (Com object)
- ScanX.dll (ActiveX object) - Mostly used for VB scripts

Note:

If you do not use the COM interface in your application and you use the SDK files directly like in VC++, then you do not need to install these files on the destination computer.

Properties and Methods

ChequeCOM Library Functions

InitChequeLibrary

Format

```
InitChequeLibrary (License As String) As Long
```

Parameters

[in] License – Null terminated string that holds the license key value.

Return Value

SLIB_ERR_SCANNER_BUSSY: The scanner is still busy executing the previous scanner command.
LICENSE_VALID: License is valid and the library is ready to be used.
LICENSE_INVALID: The license is invalid. All scanner operations are disabled.
LICENSE_EXPIRED: License has expired. All scanner operations are disabled.
SLIB_ERR_DRIVER_NOT_FOUND: The scanner driver was not found. To fix this error re-install the scanner's driver. All scanner operations are disabled.
SLIB_ERR_SCANNER_NOT_FOUND: The scanner is not connected to the PC. To fix this error make sure that the scanner is connected and re-start the function. All scanner operations are disabled.

Remarks

Use this function to initialize the **ChequeCOM** library. This function loads the scanner driver and initializes the internal image structure. This function must be called before calling any other function in the library.

CalibrateScanner

Format

```
CalibrateScanner () As Long
```

Return Value

The function returns one of the following values:
SLIB_ERR_SCANNER_BUSSY: The scanner is still busy executing the previous scanner command.
LICENSE_INVALID – Library was not initialized with the proper license.
SLIB_ERR_SCANNER_NOT_FOUND – No attached scanner was found.
SLIB_ERR_INVALID_SCANNER – The attached scanner is invalid.
SLIB_FALSE – The operation failed (Mostly because no calibration card was found).
SLIB_TRUE – Operation succeeded.

Remarks

This function calibrates the scanner using the calibration card. The calibration results are stored in a file inside the windows directory.

Clean

Format

```
Clean () As Long
```

Return Value

SLIB_ERR_SCANNER_BUSSY: The scanner is still busy executing the previous scanner command.

Remarks

This function cleans the scanner lens by running the cleaning pad (supplied in the scanner kit) back and forth. This function applies only to scanner modules ScanShell® 800/N.

ScanToFile

Format

```
ScanToFile (FileName As String) As Long
```

Parameters

[in] FileName – Null terminated string that holds the full path of the scanned image.

Return Value

If the function succeeds, the return value is **SLIB_ERR_NONE.**
If the function fails, the return value may be one of the following:
SLIB_ERR_SCANNER_BUSSY: The scanner is still busy executing the previous scanner command.
LICENSE_INVALID – Library was not initialized with the proper license.
SLIB_ERR_SCANNER_NOT_FOUND – No attached scanner was found.
SLIB_ERR_SCANNER_GENERAL_FAIL
SLIB_ERR_SCANNER_NOT_FOUND
SLIB_ERR_HARDWARE_ERROR
SLIB_ERR_PAPER_FED_ERROR
SLIB_ERR_SCANABORT
SLIB_ERR_NO_PAPER
SLIB_ERR_PAPER_JAM
SLIB_ERR_FILE_IO_ERROR
SLIB_ERR_PRINTER_PORT_USED
SLIB_ERR_OUT_OF_MEMORY

Remarks

Scan document to the internal image buffer and, at the same time, save the image into a bitmap file under the name given in the parameter "File Name". Notice that by default, the scanner is automatically set to scan the image in true color and 300 dpi for optimal OCR recognition.

After the scan, the internal image can be further manipulated and exported using separate commands such as:

- **Rotation** – Use *RotateImage*() to rotate the internal image by 90, 180 or 270 degrees.
- **Color Scheme** – Modify the internal image color to Gray or black and white images using *ReformatImage*().
- **Resolution** – Modify the internal image resolution to any resolution using *ReformatImage*().
- **Saving Format** – Save the internal image to an external file in one of 7 popular file formats using either *ConvertImage*() or *ReformatImage*() or *RotateImage*()

ScanToFileEs

Format

ScanToFileEx (FileName As String) As Long

Parameters

[in] FileName – Null terminated string that holds the full path of the scanned image.

Return Value

If the function succeeds, the return value is **SLIB_ERR_NONE.**
If the function fails, the return value may be one of the following:
SLIB_ERR_SCANNER_BUSSY: The scanner is still busy executing the previous scanner command.
LICENSE_INVALID – Library was not initialized with the proper license.
SLIB_ERR_SCANNER_NOT_FOUND – No attached scanner was found.
SLIB_ERR_SCANNER_GENERAL_FAIL
SLIB_ERR_SCANNER_NOT_FOUND
SLIB_ERR_HARDWARE_ERROR
SLIB_ERR_PAPER_FED_ERROR
SLIB_ERR_SCANABORT
SLIB_ERR_NO_PAPER
SLIB_ERR_PAPER_JAM
SLIB_ERR_FILE_IO_ERROR
SLIB_ERR_PRINTER_PORT_USED
SLIB_ERR_OUT_OF_MEMORY

Remarks

Scan document to the internal image buffer and, at the same time, save the image into a bitmap file under the name given in the parameter “File Name”. In addition, the function automatically invokes a dialog bog with a scanning progress indication.
Notice that the scanner is automatically set to scan the image in true color and 300 dpi for optimal OCR recognition.

After the scan, the internal image can be further manipulated and exported using separate commands such as:

- **Rotation** – Use *RotateImage*() to rotate the internal image by 90,180 or 270 degrees.
- **Color Scheme** – Modify the internal image color to Gray or black and white images using *ReformatImage*().
- **Resolution** – Modify the internal image resolution to any resolution using *ReformatImage*().

- **Saving Format** – Save the internal image to an external file in one of 7 popular file formats using either *ConvertImage*() or *ReformatImage*() or *RotateImage*()

ProcessCheque

Format

```
ProcessCheque (Reserved As String) As Long
```

Parameters

[in] Reserved – Null terminated empty string (“”).

Return Value

The following are the error values (negative values):

LICENSE_INVALID – Library was not initialized with the proper license.
INVALID_INTERNAL_IMAGE – The scanner image buffer is empty.
INVALID_INTERNAL_IMAGE – The scanner image buffer is empty.

If none of the above (negative values) is returned, then the image was successfully analyzed. In such a case, the function returns the angle that the image was rotated in order to be positioned in the proper alignment.
Angle_0 – The image detected to be in the proper alignment.
Angle_90 – The image was rotated by 90 degrees.
Angle_180 – The image was rotated by 180 degrees.
Angle_270 – The image was rotated by 270 degrees.

Remarks

Use this function after a check is scanned. This function takes the image from the scanner’s internal buffer, rotates it automatically to the proper alignment and analyzes the image. In return, the function returns the amount of 90 degrees rotations that the image needed to be rotated to be placed in the proper alignment. This value will help you to decide if the displayed images need to be refreshed.

GetChequeData

Format

```
GetChequeData (type As Integer, value As String) As Long
```

Parameters

[in] type – Field descriptor enum.

[out] value – The field value.

Return Value

CHEQUE_INVALID_FIELD – The value given in *type* is not a valid field descriptor.
CHEQUE_OK – Date retrieved successfully.

Remarks

Use this function after a call to ProcessCheque was made to retrieve the date printed on the check. The following field types can be extracted (the numeric value of each type is in brackets):

- **CHECK_FIELD_AMMOUNT (0):** Returns the money amount.
- **CHECK_FIELD_COMPANY (1):** Returns the issuing company name and address.
- **CHECK_FIELD_BANK (2):** Returns the issuing bank details.
- **CHECK_FIELD_MICR (3):** Returns the MICR line.
- **CHECK_FIELD_CHECK_NUM (4):** Returns the Check Number as extracted from the MICR.
- **CHECK_FIELD_ROUTING_NUM (5):** Returns the Routing number as extracted from the MICR.
- **CHECK_FIELD_ACCOUNT_NUM (6):** Returns the Account number as extracted from the MICR.

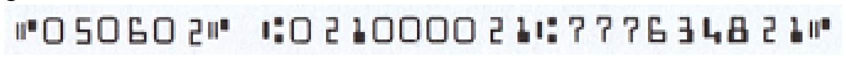
MICR Line Format

The MICR line contains special characters to identify special sections in it. These control characters are as follows:

Symbol Name	Symbol Usage	Converted to Symbol
On US ⑉	This symbol tells the reader/sorter that the next few numbers identify, among other items, the account number. Since the issuing institution determines the content of the On-Us field, the bank branch the check is drawn on may also be indicated. Look at the On-Us symbol on your personal check. Look at the numbers after it (i.e., to the left of the symbol - remember the reader/sorter reads right to left). You will recognize your account number, your bank's branch number and probably the check number. The check serial number is typically to the right of the On-Us symbol. On larger business size checks, this symbol is also used to define a field on the left end of the check. This field, called the Auxiliary On-Us field, is optional and frequently contains a multiple digit serial number.	< (0x3C)
Transit ⑆	There are always two transit symbols in a MICR line. They tell the reader/sorter that the numbers between these symbols identify the institution upon which the check is drawn (which is where the document should be sent for processing). Keep in mind that checks are not processed in branch offices, but in central processing locations, which ensures that documents take the shortest route and the shortest processing time.	: (0x3A)
Amount ⑇	The two amount symbols tell the reader/sorter that the numbers between the symbols are the amount of the check. You will seldom see this symbol, or the amount field, when you are servicing the MICR printer. This is because the amount is normally added later by the bank. However, some customer applications may add the amount field while printing checks.	; (0x3B)
Dash ⑈	The dash is sometimes used as a separator within the On-Us field, although reader/sorter manufacturers discourage its use because of detection problems. Some banks use the dash to separate the bank branch number from the account number.	= (0x3D)

Example:

The following MICR line:


⑉050602⑉ ⑆021000021⑆777634821⑉

Will be stored in the CHECK_FIELD_MICR property as:

<050602< :021000021:777634821<

ConvertImage

Format

```
ConvertImage ( _
    SourceImage As String, _ DestImage As String _ ) As Long
```

Parameters

[in] SourceImage – Full path name of the original image file. If this string is empty, the rotation is performed on the internal image.

[in] DestImage – Full path name of the destination file.

Return Value

If the function succeeds, the return value is **IMG_ERR_SUCCESS**.

If the function fails, it returns one of the following values:

- **LICENSE_INVALID** – Library was not initialized with the proper license.
- **IMG_ERR_BAD_DESTINATION** – Bad destination parameter (the destination parameter is neither file nor clipboard).
- **IMG_ERR_FILE_OPEN** – Cannot open input file. This value is returned if the SourceImage string is not empty but it points to a missing or invalid source image file.
- **INVALID_INTERNAL_IMAGE** – This value is returned if the SourceImage string is empty but no document was scanned so there is no internal image in the memory.
- **IMG_ERR_FILE_SAVE_TO_FILE** – Cannot save destination file due to an invalid destination file or disk save error.

Remarks

This function is a shorter version of the function *RotateImage*(). It takes an input file (if the SourceImage is not empty) or uses the internal image as a source (if the SourceImage is empty) and saves it to a file. Using different file extensions for the source and the destination, converts the image type to the desired type. Image type conversion must be done carefully since some color schemes are not supported in all file types. The following table shows the available destination types and the color scheme they are capable of storing:

Destination Image Extension	Destination Image Type			
	True color (24 bit)	256 colors (8 bit)	Gray scale (8 bit)	Black and white (1 bit)
BMP	√	√	√	√
TIF	√		√	√
JPG	√		√	
PCX	√	√	√	
TGA	√	√	√	
PNG	√	√	√	
PSD	√			

Important: This table is applicable to all the functions in this library.

ReformatImage

Format

```
ReformatImage ( _
    SourceImage As String, _
    toColor As Integer, _
    toDpi As Integer, _
    DestImage As String _
    ) As Long
```

Parameters

[in] SourceImage – Full path name of the original image file. If this string is empty, the rotation is performed on the internal image.

[in] toColor – One of five values:

- **LICENSE_INVALID** – Library was not initialized with the proper license.
- **IMAGE_SAME_COLOR** – No modification in the image color scheme.
- **IMAGE_BW** – Convert to black and white color scheme.
- **IMAGE_GRAY_256** – Convert to 256 gray scale color scheme.
- **IMAGE_COLOR_256** – Convert to 256-color scheme.
- **IMAGE_COLOR_TRUE** – Convert to true color scheme.

[in] toDpi – Set the new Image DPI. A value of 0 indicates no DPI modification.

[in] DestImage – Full path name of the destination file. If this value is an empty string, no save will be performed.

Return Value

If the function succeeds, the return value is **IMG_ERR_SUCCESS.**

If the function fails, it returns one of the following values:

- **IMG_ERR_BAD_COLOR** – Bad toColor parameter value.
- **IMG_ERR_BAD_DPI** – Bad toDpi parameter value.
- **IMG_ERR_FILE_OPEN –** Cannot open input file. This value is returned if the SourceImage string is not empty but it points to a missing or invalid source image file.
- **INVALID_INTERNAL_IMAGE –** This value is returned if the SourceImage string is empty but no document was scanned so there is no internal image in the memory.
- **IMG_ERR_FILE_SAVE_TO_FILE** – Cannot save destination file.
- **IMG_ERR_FILE_SAVE_TO_FILE** – Cannot save destination file due to invalid destination file or disk save error.

Remarks

Use this function to modify the image color scheme and resolution and save it to a file in any one of seven formats. The manipulated image may be loaded from an external file (if the SourceImage string holds a string value equal to the source image file name) or performed on the internal image buffer (if the SourceImage string is empty). When using a file as the image source, it is important to use the proper file extension to indicate the image format. Proper extension types are: BMP, JPG, TIFF, PCX, PNG, TGA and PSD. If an image has an unrecognizable extension due to an error (e.g. TIFF instead of TIF), the function refers to the file as BITMAP.

Image reformat can be done either on the image color scheme or the image dpi or both. Notice that changing the image format may lose the image color information (e.g., when converting from 24 bit true color to 256 gray scale). Modifying an image format from 256 gray scales to 24 bit true color will (obviously) not add color to the image but it will save the image with the proper RGB format (no color map) instead of using the 256 gray scale palette.
After the image is reformatted it can be exported to an external image file. The destination file name may be one of the seven file formats indicated above. If the destination file name has an unrecognizable extension, the function exports to the file in a BITMAP format (the default format). If no destination image file name is given (empty string), no save is done.

<u>Important</u>: The 256 colors scheme is NOT supported for JPG and TIF files.

RotateImage

Format

```
RotateImage ( _
    SourceImage As String, _
    Angle As Long, _
    DestType As Long, _
    DestImage As String _
    ) As Long
```

Parameters

[in] SourceImage – Full path name of the original image file. If this string is empty, the rotation is performed on the internal image.

[in] Angle – The angle to rotate the original image. This value can be one of the following values:

ANGLE_0: 0 degrees rotation
ANGLE_90: 90 degrees rotation
ANGLE_180: 180 degrees rotation
ANGLE_270: 270 degrees rotation

[in] DestType – The destination of the rotated image. This parameter may be one of two values:

SAVE_TO_FILE: Save the image to a file. The file name should be given in the *DestImage* parameter.
SAVE_TO_CLIPBOARD: Copy the rotated image to the clipboard.

[in] DestImage – Full path name of the destination file. This parameter is ignored if the parameter DestType is set to **SAVE_TO_CLIPBOARD**. If this value is an empty string, no save will be performed.

Return Value

If the function succeeds, the return value is **IMG_ERR_SUCCESS**.
If the function fails, it returns one of the following values:

- **LICENSE_INVALID** – Library was not initialized with the proper license.
- **IMG_ERR_BAD_ANGLE_0 –** Bad rotation parameter.
- **IMG_ERR_BAD_DESTINATION** – Bad destination parameter (the destination parameter is neither file nor clipboard)
- **IMG_ERR_FILE_OPEN –** Cannot open input file. This value is returned if the SourceImage string is not empty but it points to a missing or invalid source image file.
- **INVALID_INTERNAL_IMAGE –** This value is returned if the SourceImage string is empty but no document was scanned so there is no internal image in the memory.
- **IMG_ERR_FILE_SAVE_TO_CLIPBOARD –** Cannot save image to the clipboard due to an error.
- **IMG_ERR_FILE_SAVE_TO_FILE** – Cannot save destination file due to an invalid destination file or disk save error.

Remarks

Use this function to rotate an image by 0, 90, 180 or 270 degrees and save it to a file in any one of seven formats. The manipulated image may be loaded from an external file (if the SourceImage string holds a string value equal to the source image file name) or performed on the internal image buffer (if the SourceImage string is empty). When using a file as the image source, it is important to use the proper file extension to indicate the image format. Proper extension types are: BMP, JPG, TIFF, PCX, PNG, TGA and PSD. If an image has an unrecognizable extension due to an error (e.g. TIFF instead of TIF), the function refers to the file as BITMAP.

After the image is rotated, it can be exported to either the clipboard or to an external image file. The destination file name may be one of the seven file formats indicated above. If an image has an unrecognizable extension due to an error (e.g. TIFF instead of TIF) the function exports to the file in a BITMAP format. The destination file name may be the same as the source file name. In such a case, the new file, resulting with a rotated image, will overwrite the original file. If no destination image file name is given (empty string), no save is done.

Do not be misled by the name of this function. This function's flexibility actually allows it implicitly to do the following:

- Use the following function call to convert an image file from one type to another:
  RotateImage ("xxx.bmp", ANGLE_0, SAVE_TO_FILE, "xxx.jpg")

- Use the following function call to copy an image file to the clipboard:
  RotateImage ("xxx.bmp", ANGLE_0, SAVE_TO_CLIPBOARD, "")

- Use the following function call to rotate the internal image:
  RotateImage ("", ANGLE_0, SAVE_TO_FILE, "")

- Use the following function call to save the internal image to a file:
  RotateImage ("", ANGLE_0, SAVE_TO_FILE, "xxx.bmp")

Library Properties

PaperInTray

Type

Property.

Direction

Read only.

Remarks

Detects if a document exists in the scanner tray. This property is equal to 0 if no paper is detected in the tray or non-zero if paper is in the tray.

PressedButton

Type

Property.

Direction

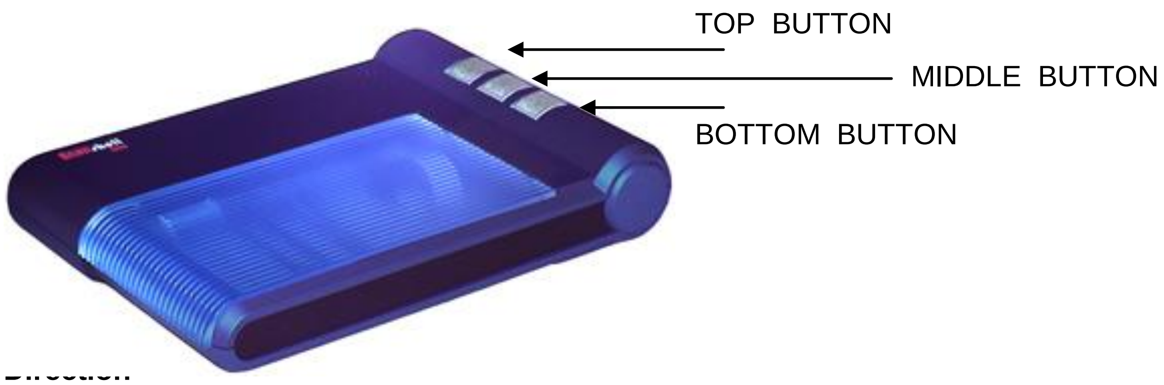
Read only.

Remarks

Returns the button number that was pressed (valid only when using the ScanShell®1000 scanner model). Read this value after reading the property *PaperInTray* (that indicates that one of the three buttons of the scanner was pressed) to retrieve which button was pressed. The return value can be one of the following:

- **TOP_BUTTON**
- **MIDDLE_BUTTON**
- **BUTTOM_ BUTTON**

These values correspond to the buttons shown in the following figure.



Direction

Read Only.

Remarks

Returns the connected scanner type. This may be one of the following values:

- CSSN_NONE
- CSSN_600
- CSSN_800
- CSSN_800N
- CSSN_1000 Not support Checks reading

- CSSN_2000
- CSSN_2000N
- CSSN_800E
- CSSN_800EN
- CSSN_3000
- CSSN_4000
- CSSN_800G
- CSSN_5000
- CSSN_IDR Not support Checks reading
- CSSN_800DX
- CSSN_800DXN
- CSSN_FDA Not support Checks reading
- CSSN_TWN Not support Checks reading

IsNeedCalibration

Type

Property.

Direction

Read Only.

Remarks

Retrieve if the scanner needs to be calibrated. This should be tested before every scan. Non-calibrated scanners may generate images with incorrect colors. The property returns a non-zero value if the scanner needs to be calibrated and a zero value if the scanner does not need to be calibrated.

IsScannerValid

Type

Property.

Direction

Read Only.

Remarks

Detect if the scanner version is supported by the current code. This property is 0 if the scanner is not supported and non-zero if it is supported.

JpegQuality

Type

Property.

Direction

Read\Write.

Remarks

This property sets\retrieves the quality factor that is used when converting images in JPEG format.
Range: 11-100 (were 11 yields the smallest image file size and 100 yields the highest image quality).

ScanProg

Type

Property.

Direction

Read.

Remarks

Use this property to retrieve the current scan progress when using the function *ScanToFileEx.* This property has values in the range of 0-100.

ScanWidth

Type

Property.

Direction

Read\Write.

Remarks

Use this property to set the scanner to the proper scan width (defaulted to 360 milli-inches, i.e., 3.6 inch).

ScanHeight

Type

Property.

Direction

Read\Write.

Remarks

Use this property set the scanner to the proper scan height (defaulted to 220 milli-inches, i.e., 2.2 inch).
Make sure to change this property to 850 (i.e., 8.5 inch) to scan the entire check document.

Note: Setting both ***ScanHeight*** and ***ScanWidth*** to –1 will automatically detect the document size. This option is useful when dealing with different document sizes.

Using ChequeCOM with ScanW

ChequeCOM SDK provides a complete and independent solution for scanning checks. This SDK can also integrate with ScanW SDK and BizCOM SDK (that analyzes driver's license, passports and business cards) and expand its capabilities to cover check reading.
In fact, both SDKs are designed to share most of their files. For example, when using both SDKs, you may control the scanner either using **ChequeCOM** (using the ScanToFile function) or using the scanning library SlibEx that is encapsulated in ScanW (using the function ScanToFile). Both SDKs will actually run the same code eventually. The only motivation to this redundancy is the will to keep **ChequeCOM** as an independent library.

To work with the two libraries (or in other words, to add a check detection capability to the ScanW library), you must install both SDKs to the same directory (ScanW first and ChequeCOM second).

Appendix A – ChequeCOM SDK Installation and Registration

Installing the ChequeCOM SDK package

The **ChequeCOM** SDK files are packed in a setup file. Installing the setup file extracts the following file list:

File Name	Destination Directory	Functionality
ChequeCOM .dll	Sdk destination directory	Library wrapper COM object
ChequeLib.dll	Sdk destination directory	Cheque analyzer.
dic.dll	Sdk destination directory	
ChecksDB.bin	Sdk destination directory	
EngRotDB.bin	Sdk destination directory	Card alignment template detection
imgForm.DLL	Windows\system32	Image processing tools collection
ImageCtrl.dll	Sdk destination directory	Image processing tools collection
Lib0.lib	Sdk destination directory	
Lib1.lib	Sdk destination directory	
Lib2.lib	Sdk destination directory	
OCR_PreProc.dll	Sdk destination directory	Driver's license image analyzer
SLib.dll	Sdk destination directory	Controls the scanner activity
SOCRdll.dll	Sdk destination directory	OCR library
test5.gar	Sdk destination directory	OCR library
test5.n3s	Sdk destination directory	OCR library
test5.qnp	Sdk destination directory	OCR library
test5.teh	Sdk destination directory	OCR library
TOCRR.ini	Sdk destination directory	OCR library
TOCRRdll.dll	Sdk destination directory	OCR library
TOCRRService.exe	Sdk destination directory	OCR library
UsaIDs.bin	Sdk destination directory	USA template collections for auto-state detection
Msvbvm50.dll	Sdk destination directory	
ATL.DLL	Windows\system32	Part of Windows system (**Installs only if newer**)
MSI.DLL	Windows\system32	Part of Windows system (**Installs only if newer**)

After the extraction, the installation program registers ATL.DLL (if newer) and MSI.DLL (if newer) and eventually registers Scanw.dll.

Manual Registration 1:

Another method to register the ScanW.dll library is to use the mouse right click button. To use this option, you must first merge the attached OCX file '**ocxdllreg.reg**' (can be found under the TOOLS directory) by clicking on it and selecting the YES button when asked.
This will enable the option to register ScanW.dll using the mouse when you click on the file name in Windows Explorer with the mouse right button.

Manual Registration 2:

An additional method to register Scanw.dll is to open a shell command prompt and in the SDK files destination directory, type the following command:

REGSVR32.EXE **ChequeCOM** .DLL

Obtaining the Full SDK package
The full SDK package includes the SDK files and documentation. VB and VC++ sample codes can be obtained from:
http://www.id-scan.com/ FTP/Applications/SDK/SDK.zip

Appendix B – Constant Definitions

The following values are used as constants:

Library SlibEx constants

```
' Scanner color scheme types
Public Const GRAY = 1
Public Const BW = 2
Public Const HT = 3
Public Const TRUECOLOR = 4

' Scanner return values
Public Const SLIB_FALSE = 0
Public Const SLIB_TRUE = 1

' Scanner general error types
Public Const SLIB_ERR_NONE = 1
Public Const SLIB_ERR_INVALID_SCANNER = -1

' Scanning failure definition
Public Const SLIB_ERR_SCANNER_GENERAL_FAIL = -2
Public Const SLIB_ERR_CANCELED_BY_USER = -3
Public Const SLIB_ERR_SCANNER_NOT_FOUND = -4
Public Const SLIB_ERR_HARDWARE_ERROR = -5
Public Const SLIB_ERR_PAPER_FED_ERROR = -6
Public Const SLIB_ERR_SCANABORT = -7
Public Const SLIB_ERR_NO_PAPER = -8
Public Const SLIB_ERR_PAPER_JAM = -9
Public Const SLIB_ERR_FILE_IO_ERROR = -10
Public Const SLIB_ERR_PRINTER_PORT_USED = -11
Public Const SLIB_ERR_OUT_OF_MEMORY = -12

Public Const SLIB_ERR_BAD_WIDTH_PARAM = -2
Public Const SLIB_ERR_BAD_HEIGHT_PARAM = -3

Public Const SLIB_ERR_BAD_PARAM = -2

Public Const SLIB_LIBRARY_ALREADY_INITIALIZED = -13
Public Const SLIB_ERR_DRIVER_NOT_FOUND = -14
Public Const SLIB_ERR_SCANNER_BUSSY  = -15
Public Const SLIB_ERR_IMAGE_CONVERSION = -16
Public Const SLIB_UNLOAD_FAILED_BAD_PARENT = -17
Public Const SLIB_NOT_INITILIZED = -18
Public Const SLIB_LIBRARY_ALREADY_USED_BY_OTHER_APP = -19
Public Const SLIB_CONFLICT_WITH_INOUTSCAN_PARAM = -20
Public Const SLIB_CONFLICT_WITH_SCAN_SIZE_PARAM = -21

' Button definition for ScanShell1000
Public Const TOP_BUTTON = 1
Public Const MIDDLE_BUTTON = 3
Public Const BOTTOM_BUTTON = 2

‘Error values for multiple devices management
Public Const SLIB_NO_SUPPORT_MULTIPLE_DEVICES = -22
Public Const SLIB_ERR_CAM_ALREADY_ASSIGNED = -23
Public Const SLIB_ERR_NO_FREE_CAM_FOUND = -24
Public Const SLIB_ERR_CAM_NOT_FOUND = -25
Public Const SLIB_ERR_CAM_NOT_ASSIGNED_TO_THIS_APP = -26
```

License related constants

```
Public Const LICENSE_VALID = 1
Public Const LICENSE_EXPIRED = -20
Public Const LICENSE_INVALID = -21
Public Const LICENSE_DOES_NOT_MATCH_LIBRARY = -22
Public Const GENERAL_ERR_PLUG_NOT_FOUND = -200
```

Library CImage constants

```
Public Const IMG_ERR_SUCCESS = 0
Public Const IMG_ERR_FILE_OPEN = -100
Public Const IMG_ERR_BAD_ANGLE_0 = -101
Public Const IMG_ERR_BAD_ANGLE_1 = -102
Public Const IMG_ERR_BAD_DESTINATION = -103
Public Const IMG_ERR_FILE_SAVE_TO_FILE = -104
Public Const IMG_ERR_FILE_SAVE_TO_CLIPBOARD = -105
Public Const IMG_ERR_FILE_OPEN_FIRST = -106
Public Const IMG_ERR_FILE_OPEN_SECOND = -107
Public Const IMG_ERR_COMB_TYPE = -108

Public Const IMG_ERR_BAD_COLOR = -130
Public Const IMG_ERR_BAD_DPI = -131
Public Const INVALID_INTERNAL_IMAGE = -132

' image saving target definition
Public Const SAVE_TO_FILE = 0
Public Const SAVE_TO_CLIPBOARD = 1

' image rotation angle definitions
Public Const ANGLE_0 = 0
Public Const ANGLE_90 = 1
Public Const ANGLE_180 = 2
Public Const ANGLE_270 = 3

' image combination options
Public Const IMAGE_COMB_VERTICAL = 0
Public Const IMAGE_COMB_HORIZONTAL = 1

' image color conversion
 Public Const IMAGE_SAME_COLOR = 0
 Public Const IMAGE_BW = 1
 Public Const IMAGE_GRAY_256 = 2
 Public Const IMAGE_COLOR_256 = 3
 Public Const IMAGE_COLOR_TRUE = 4
```

Library Check constants

```
Public Const CHECK_FIELD_AMMOUNT = 0
Public Const CHECK_FIELD_COMPANY = 1
Public Const CHECK_FIELD_BANK = 2
Public Const CHECK_FIELD_MICR = 3
Public Const CHECK_FIELD_CHECK_NUM = 4
Public Const CHECK_FIELD_ROUTING_NUM = 5
Public Const CHECK_FIELD_ACCOUNT_NUM = 6
```