**Digital Check Corp. API (JAVA)**

TellerScan™ 240,

CheXpress30

Version 1.0.0.0 (32-bit)

Windows Reference Guide

January 28, 2016

Digital Check Corporation

Copyright © 1995-2016

ALL RIGHTS RESERVED**SOFTWARE LICENSE AND DEVELOPER'S AGREEMENT**

The Digital Check Corporation Application Programming Interface (“DCCAPI”) was developed by Digital Check Corporation ("DCC") and UniSoft Imaging ("UNISOFT") to make development of applications using the SB500, SB600, SB650, SB700, BX7200, TellerScan™ 220E, TellerScan™ 230, TellerScan™ 240, TellerScan™ 4120 and CheXpress 30 check scanners easier. This software is intended for exclusive use by DCC customers or DCC resellers with scanners originally purchased from Digital Check Corporation. Use of any features in this interface is illegal with scanners not provided by Digital Check Corporation. The DCCAPI is copyrighted by Digital Check Corporation and UniSoft Imaging.

If you disagree with these terms, please return the Product and the documentation to DCC or UNISOFT, postage prepaid, within three days of you receipt, and DCC or UNISOFT will provide you with a refund, less freight and normal handling charges.

The software function libraries, programs, and related documentation (hereinafter the "SOFTWARE") are licensed, not sold. By accepting this license, you receive the right to use this SOFTWARE under the terms of this license. DCC and UNISOFT retain all rights not expressly granted you under this agreement. The SOFTWARE is trade secrets of DCC and UNISOFT and is protected by copyright, trade secret, and trademark law.

1) License Grant. DCC and UNISOFT grant you a nonexclusive right to use the SOFTWARE and documentation on the following terms:

a) You may not release any part of precompiled source files or documentation to customers or other entities without DCC and UNISOFT's written consent and proper Non‑disclosure agreements in place in a form acceptable to DCC and UNISOFT.

b) You may not sublicense, license, sell, market, assign, or transfer ownership of precompiled source files or documentation to any entity. All these rights remain with DCC and UNISOFT. This clause shall not in any way restrict your use of agents and resellers to sell your applications containing the SOFTWARE.

c) Only your full time employees or consultants who have signed valid employment agreements containing non‑disclosure provisions may work with SOFTWARE.

d) You have the right to modify SOFTWARE, but derivative works of the product are still covered by this license. This clause does not cover any application works that are completely developed by you.

e) Your applications that are developed using the SOFTWARE should only be used on check scanners originally obtained from DCC.

2) Upgrade Policy. At its option, DCC and UNISOFT may (but are not required to) make Updates and New Versions of the SOFTWARE available to you. All Updates and New Versions provided to you shall be governed by the same terms of this License Agreement.

3) Disclaimer of Warranties. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" AND WITHOUT EXPRESS OR LIMITED WARRANTY OF ANY KIND BY DCC OR UNISOFT OR ANYONE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DISTRIBUTION OF THE SOFTWARE INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE AND DOCUMENTATION IS WITH YOU. DCC AND UNISOFT SPECIFICALLY RESERVE THE RIGHT TO MAKE CHANGES OR PROVIDE MAINTENANCE OF THE SOFTWARE AND DOCUMENTATION. DCC OR UNISOFT ARE NOT RESPONSIBLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY KIND.

4) Limitation of Liability. DCC and UNISOFT's entire aggregate liability and your exclusive remedy shall be limited to the amount paid to DCC and UNISOFT for SOFTWARE and services rendered by DCC and UNISOFT under this agreement.

Table of Contents

[DCCAPI Quick Start Programing 5](#_Toc441753658)

[Hardware and Software Requirements for USB Scanners 5](#_Toc441753659)

[Hardware Requirements 5](#_Toc441753660)

[Software Requirements 5](#_Toc441753661)

[Initialization API Functions 6](#_Toc441753662)

[BuicSetParamString() 6](#_Toc441753663)

[BuicGetParamString() 7](#_Toc441753664)

[BuicInitPath() 8](#_Toc441753665)

[BuicInit() 8](#_Toc441753666)

[IsDccUsbScannerAvailable() 9](#_Toc441753667)

[Closing API Function 9](#_Toc441753668)

[BuicExit() 9](#_Toc441753669)

[Eject or Track Clearing or Calibrating API Functions 9](#_Toc441753670)

[BuicEjectDocument() 9](#_Toc441753671)

[BuicClearDocument() 10](#_Toc441753672)

[Parameter Setting and Reading API Functions 10](#_Toc441753673)

[BuicSetParam() 10](#_Toc441753674)

[BuicGetParam () 10](#_Toc441753675)

[BuicReadConfig() 27](#_Toc441753676)

[BuicWriteConfig() 27](#_Toc441753677)

[Scanning API Functions 27](#_Toc441753678)

[DccScanToFile() 27](#_Toc441753679)

[DccScanToMemory() 30](#_Toc441753680)

[DccScanSetSpecialDocument() 34](#_Toc441753681)

[DccScanSetSpecialDocumentEx() 35](#_Toc441753682)

[DccScanGetOcrMicr() 37](#_Toc441753683)

[DccScanLong() 37](#_Toc441753684)

[BuicScanGray() 38](#_Toc441753685)

[BuicScanMemoryGray() 38](#_Toc441753686)

[BuicScan() 39](#_Toc441753687)

[BuicScanMemory() 40](#_Toc441753688)

[DccScanUV() 41](#_Toc441753689)

[Endorsement API Functions 41](#_Toc441753690)

[DccBatchPrintBMP() 42](#_Toc441753691)

[DccBatchPrintString() 43](#_Toc441753692)

[DccScanVirtualEndorsement () 43](#_Toc441753693)

[Other API Functions 44](#_Toc441753694)

[BuicStatus() 45](#_Toc441753695)

[BuicStatusDelay() 45](#_Toc441753696)

[DccApiVersion() 45](#_Toc441753697)

[DccApiSupportedScanners() 45](#_Toc441753698)

[GetScannerType() 46](#_Toc441753699)

[BuicGetScannerInfo() 46](#_Toc441753700)

[BuicGetScannerSerialNumber() 47](#_Toc441753701)

[BuicCropFile() 48](#_Toc441753702)

[BuicCombineTIFFS() 48](#_Toc441753703)

[BuicDebug() 49](#_Toc441753704)

[BuicCopyFile() 49](#_Toc441753705)

[BuicCompressImage() 50](#_Toc441753706)

[BuicCompressImageGray() 50](#_Toc441753707)

[DccUVGain() 51](#_Toc441753708)

[DccUVTiff () 51](#_Toc441753709)

# DCCAPI Quick Start Programing

// Get SCAN API Object

DccScanApi scanApi = new DccScanApi();

// Set INI path

String sPath = "C:\\FullPath\\BUICSCAN.INI";

scanApi.BuicSetParamString(135,sPath);

// Initialize Scanner

result = scanApi.BuicInit();

if (result != 1) { ERROR }; // Scanner did not initialized

scanApi.BuicSetParam(160,1); // Set to batch mode

scanApi.BuicSetParam(109,0); // Set to Wait mode

// Scan images to files until feeder empty

do {

result = scanApi.DccScanToFile(sFrontTiffFileName, sBackTiffFileName,

sFrontJPEGFileName , sBackJPEGFileName, asMICR , aiFinalImageQuality, aiFinalContrast, aiDocStatus);

} while (result != -212)

// Exit Scanning Session

scanApi.BuicExit();

# Hardware and Software Requirements for USB Scanners

USB scanners

* DCC TellerScan™ 240
* CheXpress 30

## 

## Hardware Requirements

## 

* 1. GigaHz Intel/AMD Computer 4 gigabytes of Ram, Fast video card preferred

Computer

DCC Supported USB Check Scanner (TS240, CX30)

## Software Requirements

Windows 7 or higher (32/64-bit) or higher

Oracle Java version 1.8 (32-bit) or higher

DCC Teller Scan API (Buicap32.dll) version 15.12 or higher

# Initialization API Functions

## BuicSetParamString()

int BuicSetParamString(int iParameter, String sParamString);

BuicSetParamString sets parameters that are strings such as Font path and file name, firmware path and file for TS200, configuration path (to place generated files), DLL path, and location of \*.ini file.

NOTE: CFG\_INIPATH, CFG\_CFGPATH, CFG\_DLLPATH, CFG\_SCANNERTYPE and CFG\_FONTPATH must be specified before calling BuicInit. CFG\_FONTPATH can be specified before or after BuicInit. CFG\_INIPATH and CFG\_FIRMWAREPATH must specify both path and file name. CFG\_CFGPATH and CFG\_DLLPATH specify only the path. CFG\_CFGPATH must be a path with write capability.

Parameters:

|  |  |
| --- | --- |
| iParameter | Parameter to set.  CFG\_INIPATH Path to DCCAPI Ini File or “NOINI”  CFG\_CFGPATH Directory for temporary file storage  CFG\_DLLPATH Directory for TS2DLL.DLL & TS4DLL.DLL  CFG\_FIRMWAREPATH Path to firmware file  CFG\_FONTPATH Path to font File  CFG\_IJPRINTER\_FONT12FILENAME Location of ts200\_IJASCIIFont.bin file  CFG\_SCANNERTYPE 200 – Try USB Scanners First, 400–SCSI  500 or 600 for SB scanners.  CFG\_MICR\_METHOD “US” or “HTL” on USB Scanners,  “US” is recommended and necessary for  SB scanners. |
| sParamString | New value for the parameter specified in iParameter. |

Returns:

0

Examples

/\* Change default location of initialization file \*/

BuicSetParamString(CFG\_INIPATH, "NOINI"); // NO INI File

or

BuicSetParamString(CFG\_INIPATH, BuicIniFileName);

/\* Location of creating temp file \*/

BuicSetParamString(CFG\_CFGPATH, "c:\\buicrel\\ts220\\bin");

/\* Directory for TS4DLL.DLL \*/

BuicSetParamString(CFG\_DLLPATH, "c:\\buicrel\\ts220\\bin");

/\* Firmware Location \*/

BuicSetParamString(CFG\_FIRMWAREPATH, "c:\\ts220\\bin\\ts220firmware.bin");

/\* Font FileName \*/

BuicCSetParamString(CFG\_IJPRINTER\_FONT12FILENAME, "c:\\bin\\ts200\_IJASCIIFont.bin");

BuicSetParamString(CFG\_SCANNERTYPE, "200"); //Try USB Scanners First at Initialization

Recommendations:

Set CFG\_INIPATH is usually a good idea. The other settings can be very helpful when files are not stored in there default locations or when applications change the current directory while executing. If not using a configuration file, then DCC recommends using for USB Scanners:

BuicSetParamString(CFG\_INIPATH, "NOINI");

BuicSetParamString(CFG\_SCANNERTYPE, "200");

## BuicGetParamString()

String BuicGetParamString(int iParameter)

BuicGetParamString - function gets the path and/or filename (or string) for various parameters in the configuration files accessed by the scanners. Currently supported at time of original writing of function are sDLLPath, sIniFileName, sCfgPathName, sFirmwarePathName, and sFontFileName.

iParameter - Parameter to be set or should I say string.

CFG\_DLLPATH, sDLLPath - Path of the DLL's.

CFG\_INIPATH, sIniFileName - File name and path of BUICScan.ini or new name.

CFG\_CFGPATH, sCfgPathName - Path Name to write ts4cfg.cfg and ts4cfg.tmp.

RETURN:

String value of the iParameter

NULL

## BuicInitPath()

int BuicInitPath(String sDLLPath,String sIniFileName, String sCfgFileName)

BuicInitPath - function checks to see if name of program (DLL) is correct. It then checks the ASPI interface to see if a SCSI controller is out there and working.

PARAMETERS:

sDLLPath - Path of the DLL's.

sIniFileName - File name and path of BUICScan.ini or new name.

sCfgFileName - Path of ts4cfg.cfg or new name.

RETURN:

TRUE - Everything initialized correctly.

BUIC\_ERROR\_NOMANAGER - No ASPI manager present.

BUIC\_ERROR\_ILLEGALMODE - Illegal mode for ASPI.

BUIC\_ERROR\_OLDMANAGER -

BUIC\_ERROR -

NOTE: This function is Obsolete and Users should review BuicInit and

BuicSetParamString(...)

## BuicInit()

int BuicInit();

BuicInit initializes the DLL and checks the status of the SCSI/USB connection. BuicInit assumes BUICScan.ini is located in the Windows folder or Application path. BuicInit returns a 0 or positive integer (0, 1, …) if the DLL is initialized properly. A General Error Code (negative integer) is returned if a problem is encountered while querying the SCSI device driver/adapter or getting the status of the scanner. (All error codes and their meanings are listed in section Error Codes).

BuicInitPath is no longer recommended. It is recommended that user use BuicSetParamString instead of BuicInitPath.

## IsDccUsbScannerAvailable()

int IsDccUsbScannerAvailable(int[] aiDCCProductVID, int[] aiDCCProductPID)

IsDccUsbScannerAvailable() function tests the USB, universal serial bus, for a valid Digital Check Scanner. The VID, Vendor Id, should be 0x08b1 and the PID, Produce Id, reflects the model number. The return code is the actual model number if possible.

RETURN:

0 - No DCC scanner found on USB bus available OR procedure failed.

1 - DCC scanner found but not sure which

CX30 - CX30 found

TS220 - TS220 Found

TS230 - TS230 Found

TS240 - TS240 Found

BX7200 - BX7200 Found

SB500 - SB500 Found

SB600 - SB600 Found

# Closing API Function

## BuicExit()

int BuicExit()

BuicExit frees resources and memory used by the DLL. BuicExit should be called before exiting an application or before calling BuicInit a second time to free all resources and memory. BuicExit returns a 0 or positive integer if successful or a General Error Code otherwise.

# Eject or Track Clearing or Calibrating API Functions

## BuicEjectDocument()

int BuicEjectDocument()

This function ejects a halted document from check scanners. It returns either 0 or EJECT\_ERR. DCC highly recommends the user add an Eject button to clear the scanner track, since pulling documents out is hard on the wheels of the scanner track.

## BuicClearDocument()

int BuicClearDocument()

When using SCANBATCH mode, the user might want to clear the next document scanned. This function will return -212, SCAN\_NO\_CHEQUES, if a document had not been pre-scanned and -217, SCAN\_DOUBLE\_FEED, if a document had been pre-scanned.

# Parameter Setting and Reading API Functions

## BuicSetParam()

int BuicSetParam(int iParam, int iValue);

BuicSetParam will set the configuration parameter to the specified value. BuicSetParam returns a 0 or positive integer if successful or a General Error Code otherwise. See the DCCAPI Parameters section for a list of valid parameters.

Parameters:

|  |  |
| --- | --- |
| iParam | Parameter to be set.  See DCCAPI Parameters section for more information. |
| iValue | Value to set parameter selected in iParam. |

## BuicGetParam ()

int BuicGetParam(int iParam)

BuicGetParam will return the configuration parameter of the specified value. See the DCCAPI Parameters section for a list of valid parameters.

Parameters:

|  |  |
| --- | --- |
| iParameters  Returns: Current value of parameter select in iParameters | Parameter to be returned.  See BUIC Parameters section for more information |

These mnemonics will be used with BuicSetParam() and BuicGetParam() functions. The information will be displayed in the following manner: the mnemonic, its numeric equivalent, and the scanners that support this parameter, a list of valid values, a brief explanation, and the default value. For more information, see the configuration file section.

**CFG\_MICR\_ENABLE or BPARAM\_MAGNREADER** 1 (DCC SUPPORTED SCANNER)

BUIC\_DEV\_OFF 0 (Disable or turn off Magnetic Reader)

BUIC\_DEV\_ON 1 (Enable or turn on Magnetic Reader)

This parameter enables or disables the magnetic code or MICR reader. Default value is BUIC\_DEV\_OFF or MICR reader disabled.

**NOTE:** *This parameter is overridden by the Image Scan Format in the Scan functions.*

**CFG\_MICR\_FONT or BPARAM\_MAGNTYPE**  2 (DCC SUPPORTED SCANNER)

BUIC\_CMC7 0 (CMC7 MICR font)

BUIC\_E13B 1 (E13B MICR font)

BUIC\_OCRA 2 (Future - TellerScan™ 400 Only)

BUIC\_OCRB 3 (Future - TellerScan™ 400 Only)

This parameter defines the type or font of magnetic code (MICR) line to read. Default value is BUIC\_E13B.

**CFG\_DEV\_ENDORSER or BPARAM\_ENDORSER**  4 (DCC SUPPORTED SCANNER)

BUIC\_DEV\_OFF 0 (Disable or turn off Endorser)

BUIC\_DEV\_ON 1 (Enable or turn on Endorser)

The endorser is an optional roller stamp for the front of documents and must be ordered at time of purchase of the scanner. Default value is BUIC\_DEV\_OFF or endorser disabled.

**CFG\_IMAGE\_RESOLUTION or BPARAM\_DPI** 6 (DCC SUPPORTED SCANNER)

CFG\_IMAGE\_RESOL\_100X100 0 (100 x 100 DPI)

CFG\_IMAGE\_RESOL\_200X200 1 (200 x 200 DPI)

CFG\_IMAGE\_RESOL\_200X100RAW 2 (200 x 100 DPI – SCSI Only)

CFG\_IMAGE\_RESOL\_200X100 3 (200 x 100 DPI – SCSI Only)

CFG\_IMAGE\_RESOL\_300X300 4 (300 x 300 DPI – CX30, TS240, SB500, SB600

CFG\_IMAGE\_RESOL\_600x600 5 (600 x 600 DPI – SB500 and SB600 Only

This parameter defines the DPI of the scan. Default value is CFG\_IMAGE\_RESOL\_200X200.

**CFG\_DEV\_DOUBLE\_FEED 7**

**BPARAM\_PHOTOCELL** 7 (DCC SUPPORTED SCANNER)

BUIC\_DEV\_OFF 0 Double feed detect off or disabled

BUIC\_DEV\_ON 1 Double feed detect on or CF\_DEV\_DF\_CONTINUE 2 Double feed detected and document fed into

output tray. If MICR is bad or questionable,

a double feed error is returned otherwise no

error is returned and scanning continues as if

no double feed was detected.

(TS220, TS230, TS4120 ONLY)

Double feed detection enabled or disabled.

**BPARAM\_WINDOWX1**  19 (B1000, B1500, TS400)

**BPARAM\_IMGRIGHT**  19

Valid range is 0 to 2000 (B1000)

Valid range is 0 to 1000 (B1500, TS400)

Default value is 0.

**Note:** *The image is scanned from right to left assuming a 90 degree rotation. Therefore, the point 0,0 on the image is the bottom right corner NOT the top left corner. Consequently, Image Right must be less than Image Left, and Image Bottom must be less than Image Top.*

**BPARAM\_WINDOWY1**  20 (B1000, B1500, TS400)

**BPARAM\_IMGBOTTOM**  20

Valid range is 0 to 1200 (B1000)

Valid range is 0 to 863 (B1500, TS400)

Default value is 0.

**Note:** *The image is scanned from right to left assuming a 90 degree rotation. Therefore, the point 0,0 on the image is the bottom right corner NOT the top left corner. Consequently, Image Right must be less than Image Left, and Image Bottom must be less than Image Top.*

**BPARAM\_WINDOWX2** 21 (B1000, B1500, TS400)

**BPARAM\_IMGLEFT**  21

Valid range is 0 to 2000 (B1000)

Valid range is 0 to 1000 (B1500, TS400)

Default value is 800. (B1000)

Default value is 1000. (B1500 and TS400)

**Note:** *The image is scanned from right to left assuming a 90 degree rotation. Therefore, the point 0,0 on the image is the bottom right corner NOT the top left corner. Consequently, Image Right must be less than Image Left, and Image Bottom must be less than Image Top.*

**BPARAM\_WINDOWY2**  22 (B1000, B1500, TS400)

**BPARAM\_IMGTOP**  22

Valid range is 0 to 1200 (B1000)

Valid range is 0 to 863 (B1500, TS400)

Default value is 832. (B1000)

Default value is 863. (B1500, TS400)

**Note:** *The image is scanned from right to left assuming a 90 degree rotation. Therefore, the point 0,0 on the image is the bottom right corner NOT the top left corner. Consequently, Image Right must be less than Image Left, and Image Bottom must be less than Image Top.*

**CFG\_IMAGE\_BW**  23 (B1000, B1500, TS400)

BUIC\_MONO 0 Black/White, Bitonal, 1-bit/pixel

BUIC\_GRAY 1 Gray scale

Select whether to scan images black/white (bitonal) or gray scale. This parameter is overridden by the Scan routine parameter.

**CFG\_SETUP\_PHOTODELAY**  24 (B1000)

Valid values are 0 and 1

25-30 Reserved….

**CFG\_DEV\_PRINTER** 31

1. Disabled
2. Enabled

**CFG\_DEV\_SORTER** 32

1. Use Default Pocket

1 Use sorter if available

**CFG\_IMAGECROPPING\_ENABLE**  36 (TS400)

0 Crop using Window X1, X2, Y1, Y2

1 Crop using Window X1, X2, Y1, Y2

and then remove any black border at top of image.

Select whether to crop black border at top of documents.

Default is 0.

**CFG\_SETUP\_ROTATION**  37 (B1500, TS400)

0 No rotation, image of document on side

1 Rotation, image is horizontal

If no rotation is selected, document scanned from right edge will be rotated 90 degrees. If rotation is selected, document image will be rotated 90 to look as expected.

Default is 1.

**CFG\_IMAGE\_GRAY256LEVEL** 45 (B1500, TS400)

CFG\_IMAGE\_GRAY256LEVEL\_16 0 4-bits per pixel, 16 shades of gray

CFG\_IMAGE\_GRAY256LEVEL\_256 1 8-bits per pixel, 256 shades of gray

If gray scale is selected, this parameter selects between shades 16 and 256 shades of gray.

Default is 256 shades of gray. USB scanners should use 8. SCSI scanner might want to use 4 since it is faster.

**CFG\_SORTER\_INPUT** 49 (TS400)

BUIC\_INT 0 (Internal – scanner does sorting)

BUIC\_EXT 1 (External – application does sorting)

This parameter defines the input for the sorting routine. If BUIC\_INT is selected and sorter is turned on, the scanner sorts according to the selected algorithm. If external is selected, the application must call TS400SetPocket to specify the output pocket.

Default value is BUIC\_INT.

CFG\_SETUP\_DOC\_THICKNESS 50

See **CFG\_DEV\_DOUBLE\_FEED** (Number 7)

**CFG\_MICR\_FORMAT** 57 (BUIC1500, TS400)

MICR\_FORMAT\_NOSPACE 0 (MICR with removed spaces)

MICR\_FORMAT\_ALL\_SPACES 2 (MICR with spaces)

3 (MICR with spaces padded into Amount Field)

This parameter selects the internal magnetic reader format.

Default value is BUIC\_DEV\_ON.

**CFG\_SETUP\_FILENAMETERM** 104

0 – Spaces in Filenames are treated as termination characters

1 - Spaced are just spaces in directories or filenames

Default is 0 which is a problem on systems supporting spaces.

**CFG\_SETUP\_TIFFMICRTAG** 105

0 – off

1 – Added a private Tiff Tag 0x872c or 34602 to hold MICR string. Must use BuicUpdateMicrLine

2– Create a minimum Tiff Header of 138 bytes….

**CFG\_SETUP\_BMPLOADONCE** 106

0 – Reload BMP every time

1 – Load once – assume static endorsement

**CFG\_IMAGECROPPING\_THRESH** 108 (TS200)

This parameter selects the crop threshold for removing the top black area (overscan area) for documents. Pure black is 0 while pure white is 255. Shades of gray vary between pure black and pure white.

Default value is 80.

**CFG\_MISC\_IMAGEWAIT** 109 (TS200, TS205, TS210, TS220, TS230)

This parameter selects the number of milliseconds to wait after a BuicScan before returning –212 for no checks present. The TS200 is a drop feed and BuicScan may be issued before the document is dropped; therefore, the system will wait before returning an error of no documents present.

Default value is 10000 or 10 seconds

**TPARAM\_JPEG\_QUALITY** 110 (All Scanners)

This parameter selects the quality of JPEG image. As the JPEG quality is increased, the size of the stored image increases. Therefore, the user can decide the appropriate amount of quality versus size of image. Values such as 25 are of lower quality but are often adequate for many applications and result in smaller size images in storage.

Default value is 75.

**CFG\_MISC\_ENABLESCAN** 111 (TS200, TS205, TS210, TS220, TS230)

BUIC\_DEV\_OFF 0 (Scanner is not enabled until a BUICScan is issued)

BUIC\_DEV\_ON 1 (Scanner is re-enabled after each document is read, thus is ready to scan the next document even before BuicScan is issued)

This parameter re-enables the scanner after each document is scanned. This parameter is automatically disabled when CFG\_MISC\_SCANBATCH\_ENABLED is set.

Default value is 0, don’t re-enable scan.

**CFG\_MISC\_MICR\_VERIFY** 116 (TS400, TS200)

This parameter selects whether to use OCR verification techniques on the MICR to improve the accuracy of the MICR read by the scanner. If 0 is selected, OCR verification of the MICR is not attempted. Otherwise, the value is the number of @’s in the scanner MICR before aborting OCR MICR verification.

Default value is 4. If using DCCScan, then disabling will increase speed a little.

**CFG\_MISC\_MICR\_VERIFY\_LOG** 117 (All Scanners)

This parameter should be set to 0 unless instructed by a Technical Support person. If thevalue is set to 1, a log file of OCR verification is created which can get quite large after frequent scanning. If the value is set to 2, magnetic and OCR MICR and confidences are stored in c:\buicapi.log.

Default value is 0.

**CFG\_MISC\_MICR\_GRAY\_THRESHOLD** 118 (All Scanners)

This parameter is a multiple of 16 from 32 to 160. This specifies the threshold to use for converting grayscale images to black/white for enhanced MICR processing. The default is 96 and a value of 64 would be used on darker images and a value of 128 would be used on lighter images or document without background in the MICR area of a check.

Default value is 64.

**CFG\_IMAGE\_FRONTCOLOR** 119 (TS205 and TS220 and TS230)

BUIC\_DEV\_OFF 0 Scan in Grayscale (Default)

BUIC\_DEV\_ON 1 Scan in Color

2 Scan in slow color on TS230 and TS4120

CFG\_IMAGE\_GRAYPLUSUV 3 TS240 UV ONLY Grayscale image with

white light + UV (ONE IMAGE)

CFG\_IMAGE\_UVONLY 4 TS240 UV ONLY Grayscale UV image ONLY

CFG\_IMAGE\_DUALGRAYUV 5 TS240 UV ONLY TWO IMAGES: Grayscale

document image and UV image

**CFG\_IMAGE\_BACKCOLOR** 126 (TS205 and TS220 and TS230)

BUIC\_DEV\_OFF 0 Scan in Grayscale (Default)

BUIC\_DEV\_ON 1 Scan in Color

2 Scan in slow color on TS230 and TS4120

**CFG\_MISC\_DOUBLEFEEDDELAY** 133

30 to 50 Having to do with the sensitively of the Double Feed Sensor.

**CFG\_MISC\_PH\_WHITE\_THRESH** 143

Value used to detect the end of a document on a TS210/TS220. The default value is 40, but 30 will work for most scanners also. Set too low and the user will see extra jam errors.

**CFG\_MISC\_PH\_WHITE\_TOLER** 144

How much tolerance to allow in the white threshold setting. Default is 15.

**CFG\_MISC\_PH\_ENABLE** 145

BUIC\_DEV\_OFF 0 Disable normal Endo of Document Detection

BUIC\_DEV\_ON 1 Normal End of Document Detection using

White Threshold out Setting See 143.

**CFG\_DCCSCAN\_IQATESTS** 146

See IQAStatus Function Call variable iIQATests

**CFG\_DCCSCAN\_STARTCONTRAST** 147

Valid range is 300 to 1000 with the default usually 450

**CFG\_DCCSCAN\_ENDCONTRAST** 148

Valid range is 300 to 1000 with a default of 750

**CFG\_DCCSCAN\_INCCONTRAST** 149

Usually 100 to difference of **CFG\_DCCSCAN\_ENDCONTRAST - CFG\_DCCSCAN\_STARTCONTRAST**

**CFG\_JPEG\_IMAGE\_RESOLUTION** 150

100 or 200 dpi or 0 dpi (no Image)

**CFG\_DCCSCAN\_MAXSPECKLES** 151

Application dependent with standard Checks 1200 to 2000

**CFG\_DCCSCAN\_MINMICRQUALITY** 152

MICR quality is measured between 1 and 10 with 5 and up being valid.

**CFG\_DCCSCAN\_MAXBENTCORNER** 153

Number of pixels in a bent corner 60 to 200

**CFG\_DCCSCAN\_MAXRAWDENSITY** 154

Usually 26 and up, but all business checks could be lower

**CFG\_DCCSCAN\_MAXTIFFIMAGESIZE** 155

Maximum allowable Tiff Image Size in Kilobytes.

**CFG\_DCCSCAN\_OPTIONS**  156

Additional Options that can be turned on like:

1 - try multiple EdgeDetection Contrast

2 - try simple thresholding

4 - despeckle the images

8 – save grayscale or color images in BMP format

16-Use Group 4 Filter for smaller Tiff Images

32-Return a different thresholded Tiff Images instead of a JPEG image,

so user can pick the best on

64- - Force EdgeDetect Images to 3.1% Density (Federal Reserve Minimum

Image Density Standard)

128-Return grayscale/color as uncompressed image with a BUICMemHdr Header

256-Return Rear Image Density in iDocStatus[31] and Endorsement Characters in

iDocStatus[27]

512-Reduce 300 dpi to 240 dpi on a CX30

1024-Upside down check scanning

2048-On CX30 Only, Process MICR forward and Backward to attempt to read very low

intensity MICR

4096-Reduce 300 dpi scanning to 200 dpi. This is necessary if doing CMC7 OCR

(which only works at 300 dpi) and application needs 200 dpi images as output.

8192-Use a faster simple threshold on the rear image. Some speed increase.

16384-Test Reverse Edge for MICR (SB500 and SB600 Only)

32768-Cut DPI 600 DPI to 200 DPI (SB500 and SB600 Only)

65536-Avoid "VOIDS" on Business Checks

131072-Second Pass MICR OCR

**CFG\_MISC\_MICR\_LOWCONFIDENCE** 157

0 – If Magnetic MICR and OCR MICR conflict, use magnetic MICR results

1 – If Magnetic MICR and OCR MICR Conflict, use an @ character

**CFG\_MISC\_SCANBATCH\_ENABLE** 160

NOTE: Valid only on TS230-65 and TS230-100

0 – Do not scan ahead, i.e. do not feed in next check while processing current check.

1 – To increase throughput, scan ahead by feeding in the next document while processing the

current document.

**CFG\_MISC\_FEED\_SOLENOID**  161

NOTE: Do not change this parameter unless asked to do so by DCC personnel as it will put undue stress on solenoid motor and make it run hot.

0 – Feed Solenoid mode. Motor turned off between documents.

1 – Feed Solenoid Mode. Motor left on between documents.

**CFG\_MISC\_DEBUG\_MESSAGES** 162

NOTE: Valid only on TS230-65 and TS230-100

0 – Debug Messages OFF. Default and should always be OFF unless a member of DCC asks

you to turn it on for debug purposes.

1. – Debug Messages ON. Will slow system down, but useful for debugging.

**CFG\_MISC\_TS230\_65** 163

0 – Normal default

65 – If you have a TS230-100, setting 65 will make the

Scanner run like a TS230-65.

**CFG\_MISC\_DELAY\_TS200** 165

0 – No Debug String Output or Delay

1 – Some Debug Output and Delay

2 – Significant Debug Output and Delay

**CFG\_MISC\_SCANNER\_TYPE** 167 **// Read Only**

Real Scanner Type (USB Scanners)

**CFG\_MISC\_USB1** 168

0 – Use delays normal for USB 2.0 support

1 – Use longer delays to support USB 1.1

**CFG\_MISC\_TS200\_CK\_FED** 169

0 – No Check prefed

1 – Check was prefed because of CFG\_SCANBATCH\_ENABLE or CFG\_SCAN\_ENABLE. Note: calling this function on a TS4120 will slow down scanning, so it should only be used to quit scanning.

**CFG\_MISC\_VIRTUAL** 170

0 – Virtual Endorsement Disabled

1 – Virtual Endorsement Enabled

**CFG\_MISC\_RED\_FILTER** 171

0 – Off

1 – Right Bottom Corner

2 – Whole Check

CFG\_MISC\_USER1 172

CFG\_MISC\_USER2 173

CFG\_MISC\_USER3 174

CFG\_MISC\_USER4 175

CFG\_MISC\_USER5 176

User settable and readable

**CFG\_SCANNER\_DISABLED** 178

0 – nothing

1 – If in SCANBATCH Mode, one can disable the scanner in

Order to switch endorsements or scanning modes.

Use only when recommended by DCCAPI technical personal. This is likely to be misused otherwise.

**CFG\_AMERICAN\_MICR\_RULES** 179 – If 1 do additional tests for Double Feed No Stop checking for a single routing field and amount field if present.

**CFG\_REPORT\_MICR\_PARSE\_ERROR** 180-If 1, return an error -227, MICR\_PARSE\_ERROR, if two routing or amount fields are found on the MICR Line

**CFG\_MISC\_ADDITIONAL\_BLINK** 181- 0 to 2000. Number of additional milliseconds to wait while scanner red light is blinking on a scanner detected error.

**CFG\_FEED\_MSG\_BEEP** 183-0 off, 1 – Force a MessageBeep(1) when checks are

Loaded into the input feeder of the scanner.

**CFG\_MISC\_FORCE\_EJECT** 184- 0 off, 1 – Force application to use a BuicEject to clear a Jam. This option also leaves the input light on the scanner flashing red until cleared.

**CFG\_MISC\_LATE\_JAM\_TEST** 185 0 – off, 1 - Test during pocketing if a jam occurred around the top of the TS4120. This is a very rare event and is really not necessary.

**CFG\_REMOTE\_MONITOR\_LOG** 186 0- off, 1 – create a remote monitoring log file c:\TempPath\buicapi.txt

**CFG\_MISC\_MICR\_DROP\_SKIP** 187 0- 0ff (Default), 1 - Don’t return @@@ when magnetic MICR signal drops out.

**CFG\_REMOTE\_MONITOR\_LOG\_SIZE** 188 0-disabled or max size of remote logging file

**CFG\_IMAGE\_PADBOTTOM** 189 Extra White Scanlines at bottom. Default: 0

**CFG\_IMAGE\_PADTOP** 190 Extra White Scanlines at top. Default: 0

**CFG\_SCAN\_MODE** 191 // FOR CX30 ONLY:

**CFG\_SCAN\_MODE\_FORWARD** 0 - Scan forward direction place check in

OUTPUT tray

**CFG\_SCAN\_MODE\_REVERSE** 1 - Scan forward direction, return check in

INPUT tray

**CFG\_SCAN\_MODE\_HOLD** 2 - Scan forward direction, HOLD check after

reading.

NEXT SCAN will place check in tray.

**CFG\_DCCSCAN\_MINLENGTH** 192 -- Check Minimum Length in 10th inches.. Default is 30 for 3 inches.

**CFG\_MICR\_GAIN** 193 // FOR CX30 ONLY:

**CFG\_MICR\_GAIN\_DEFAULT** 100 – 100% Manual MICR Gain Setting only active

CFG\_MICR\_AGC is zero

**CFG\_MICR\_AGC** 194 // FOR CX30 ONLY:

0 – Use manual gain setting – CFG\_MICR\_GAIN

1-Allow DCCAPI to calculate next setting based on previous scan. Changes Default only in CFG\_SCAN\_MODE\_HOLD

**CFG\_MISC\_LATE\_JAM\_TEST\_CX30** 195 // 0 – default, 1 – Disable a late jam test in Scan & Return mode on the CX30. Version 9.03 Update

**CFG\_FRANK\_MODE** 196

CFG\_FRANK\_MODE\_OFF 0 NO FRANKING on any document - default

CFG\_FRANK\_MODE\_ON 1 ALWAYS FRANKING, every document

CFG\_FRANK\_MODE\_MICR 2 FRANK if the MICR is good

CFG\_FRANK\_MODE\_EXTERNAL 3 USER selects whether to Frank or not by calling DCCSetFrank. NOTE: in this mode, the scanner will hold the document UNTIL it receives the command DCCSetFrank.

**CFG\_FRANK\_BAD\_QTY** 197 Reserved

**CFG\_FRANK\_RIGHT\_MARGIN** 198 Reserved

**CFG\_FRANK\_LENGTH** 199 Reserved

**CFG\_DCCSCAN\_REAR\_THRESHOLD** 200 Rear threshold defaults to a contrast of 600, and this variable can override it with a higher contrast of 400 (dirty images).

**CFG\_FIRMW\_LOAD\_MODE** 201 On a TS240 the firmware is preloaded so initialization is faster.

0 – No Load

1 – Load Firmware Always

2 – Load Firmware is newer (Default)

**CFG\_DIAG\_DIALOG\_MODE** 203 /\* Control Debug Dialog \*/

#define CFG\_DIAG\_DIALOG\_MODE\_ALWAYS 1 /\* Always popup Debug Dialog-Testing Only \*/

#define CFG\_DIAG\_DIALOG\_MODE\_ARMED 2 /\* show the dialog only if armed with Alt Shift Key and Scanner Reset\*/

#define CFG\_DIAG\_DIALOG\_MODE\_SH\_FOCUS 6 /\* (Default) Show the Dialog on Shift and Focus \*/

// Read Only Informational Values about Attached Scanner 10.03

**CFG\_RO\_300DPI\_ABLE** 204 /\* 1 - 300 DPI Supported, 0 - No \*/

**CFG\_RO\_AUTO\_FEEDER** 205 /\* 1 - AutoFeeder, 0 - DropFeeder \*/

**CFG\_RO\_SCANBATCH\_ABLE** 206 /\* 1 - ScanBatch Available, 0 - no \*/

**CFG\_RO\_PRINT\_CARTRIDGE\_LOADED** 207 /\* 1 - Carriage Installed, 0 - no \*/

**CFG\_RO\_SORTER\_ABLE** 208 /\* 1 - Sorter Available, 0 - no \*/

**CFG\_RO\_DIR\_ENDORSEMENT** 209 /\* 1 - Right Side Feed, 0 - Left Side Feed \*/

**CFG\_RO\_FRANKER\_INSTALLED** 210 /\* 0 - No Franker, 1 - Franker \*/

**CFG\_RO\_DOCS\_IN\_TRACK** 211 /\* When Batch scanning, max number of documents in track \*/

**CFG\_DCCSCAN\_FIX\_IMAGES** 212 /\* Number of K bytes to force image filtering \*/

**CFG\_NONCRITICAL\_TIMEOUT** 214 /\* If enabled, test for Thread lock outs. \*/

**CFG\_PRINTER\_HEAD** 215 /\* Test whether a 4 line print head is instead. \*/

**CFG\_CMC7OCR\_BLACK** 216 /\* Default 96 – Valid Range 48 to 160 – Background grayscale level – should only be lower in countries with very dark backgrounds on their checks. This can help CMC7 OCR rates in Brazil. On SB500 and SB650, a higher value of 144 seems to help in capture of CMC7 OCR \*/

**CFG\_DEV\_USDOUBLEFEED** 217 /\* New Sensors for BX7200, should be left enabled (1) since the ulta-sonic sensor is much better \*/

**CFG\_SORT\_MODE** 218 /\* Sort mode for BX7200 \*/

CFG\_SORT\_MODE\_NO\_CHANGE 0 /\* Stay in current position \*/

CFG\_SORT\_MODE\_A\_NOW 1 /\* All documents forced to Outside Pocket \*/

CFG\_SORT\_MODE\_B\_NOW 2 /\* All documents forced to inside Pocket \*/

CFG\_SORT\_MODE\_FLIPNEXT 5 /\* Alternate pockets \*/

CFG\_SORT\_MODE\_FLIPIFFULL 6 /\* CASCADE, ROLLING pocket. Switch Pocket whenever the pocket gets full. Default non Sorting Mode \*/

CFG\_SORT\_MODE\_WAITTOSORT 8 /\* Wait for sort decision, DEFAULT Sorting Mode \*/

CFG\_SORT\_MODE\_DOCSORT 9 /\* Like 8 above, but defaults to Pocket A or 0 if sort decision is not received in time. \*/

CFG\_SORT\_MODE\_STOPTOSORT 10 /\* Wait for sort decision, but run one check in the bx7200 track at a time. VERY SLOW. \*/

**CFG\_MICR\_RESAMPLE** 219 /\* Increase Gain - retest MICR, Default on or 1 \*/

**CFG\_PRINTER\_ECONOMY\_MODE** 220 /\* Only supports on TS240 4 Line Printer \*/

/\* 0= default: Column 1 & 2 = 600DPI \*/

/\* 1= Same as Mode 0 \*/

/\* 2= Column 3 & 4 = 600DPI \*/

/\* 3= Column 1 - 4 = 600 DPI redundant printing \*/

/\* 4= Column 1 = 300DPI \*/

/\* 5= Column 3 = 300 DPI \*/

**CFG\_IMAGE\_FRONT\_UV\_THRESH** 221 // 0 to 14 start with 13 – Tiff UV Threshold \*/

**CFG\_CLEAR\_HALT\_BX7200** 222 // Calling this function causes User Halt on the

**//** BX7200 to be cleared

**CFG\_BX7200\_DF\_NOIMAGES** 223 // If DF on BX7200, don't return images

**CFG\_RO\_SCANNER\_STATUS** 224 // Actual Scanner Status - Snapshot --

// Many of these status are handle immediately

// so users can not depend on some of them.

// Valid ones are marked by star.

// BX7200, CX30, TS240 Only

CFG\_RO\_CMDCOMPLETE 0x00000001 // Can mean a scan or eject complete

CFG\_RO\_POWERON 0x00000002 // Scanner Power On Flag is Set

CFG\_RO\_SCANNING 0x00000004 // Scanner Active Scanning Mode

CFG\_RO\_ERROR 0x00000008 // Current Error Exists

CFG\_RO\_PHOTOFEEDER 0x00000010 // Documents in Track - Not valid always

CFG\_RO\_PHOTOSYNC 0x00000020 // A photo sensor is busy -- JAM if track is stopped

CFG\_RO\_PRTHEAD 0x00000040 //\*Print Cartridge is installed

CFG\_RO\_SCANENABLE 0x00000080 // Scanner Active or waiting on check

CFG\_RO\_RDTRACK 0x00000100 // Not Used

CFG\_RO\_BUSY 0x00000200 // Scanner Busy

CFG\_RO\_DOUBLEFEED 0x00000400 // Double Feed Sensor Active

CFG\_RO\_PHOTOJAM 0x00000800 // Document in optics

CFG\_RO\_INTERLOCK 0x00001000 // Scanner Door Open -- UnSafe to Scan

CFG\_RO\_MODESEL 0x00002000 // CX30 Door Closed

CFG\_RO\_PHOTOPOCKET 0x00004000 // Not Used

CFG\_RO\_POCKET\_FULL\_0 0x00008000 //\*BX7200 Pocket 0 full

CFG\_RO\_POCKET\_FULL\_1 0x00010000 //\*BX7200 Pocket 1 full

CFG\_RO\_INTRACK 0x00020000 //\*CX30 Doc InTrack

CFG\_RO\_RESERVED1 0x00040000 // unused

CFG\_RO\_RESERVED2 0x00080000 // unused

CFG\_RO\_MICRRDY 0x00100000 // MICR Signal Ready

CFG\_RO\_IMGFRDY 0x00200000 // Front Image Ready

CFG\_RO\_IMGRRDY 0x00400000 // Rear Image Ready

CFG\_RO\_RS232RDY 0x00800000 // unused

CFG\_RO\_RDYFOREARLYSCAN 0x01000000 // Scanner Ready to Start Next Doc

CFG\_RO\_DOCINBUFFER 0x02000000 // Not Used

CFG\_RO\_WAITINGSORTDEST 0x04000000 // Waiting for Pocket

CFG\_RO\_EARLYMICRRDY 0x08000000 // Early MICR Ready

CFG\_RO\_USERHALTED 0x10000000 // BX7200 User Halted Button Pressed

CFG\_RO\_RESERVED3 0x20000000 // unused

CFG\_RO\_USERSTART 0x40000000 // BX7200 Start Button Pressed

CFG\_RO\_DEVCONNECTED 0x80000000 // Scanner Connected

**CFG\_SOFT\_IMAGETOOLONG** 225 // BX7200 Software Image Too Long Test

// 0 - Disable, 1 - Enable

**CFG\_SKIP\_4L\_DELAY** 226 // 0 - Default - Delay 3 seconds

// on BuicInit so 4L cartridge

// can heat up.

// 1 - No Delay

**CFG\_DF\_ENDORSE** 227 // 0 - Default, On BX7200 don't

// endorse on DF items, 1 - Endorse on

// Double Feed items

**CFG\_RO\_DCCSCAN\_MICR\_CHG** 228 // 0 - No, 1 - Magnetic MICR Updated

// by MICR OCR (Valid only on BX7200

// when sorting

**CFG\_RO\_SORT\_POCKET** 229 // 0 - Unknown, 1 - Outside, 2 – Inside

// (Valid only on BX7200 when sorting)

**CFG\_RO\_UV\_ABLE** 230 // 0 - Scanner does not support UltraVialet

// 1 - UV Scanner

**CFG\_PLUNGERPARKTICKS** 311 /\* BX7200: Determines size of Feeder gap on BX7200\*/

Gap 0% = 10

Gap 20% = 13

Gap 40% = 16

Gap 60% = 18

Gap 80% = 21

Gap 100% = 24

## BuicReadConfig()

int BuicReadConfig(String sFilename)

BuicReadConfig - Read the configuration file whose name was passed to this function. Set up the parameter block using the values defined in the configuration file.

PARAMETERS:

sFileName - Name of the configuration file and path.

RETURN:

1 - File found and downloaded.

2 - File not found going to defaults.

BUIC\_ERROR\_BADFILENAME - File not found or at least unreadable.

E\_NO\_SCANNER - No scanner available or not yet initialized.

## BuicWriteConfig()

int BuicWriteConfig(String sFilename)

BuicWriteConfig - Write a configuration file using the values in the parameter block.

PARAMETERS:

sFileName - Name of the configuration file and path.

RETURN:

TRUE - Everything okay.

BUIC\_ERROR\_BADFILENAME - File not found or at least unwritable.

# Scanning API Functions

## DccScanToFile()

int DccScanToFile(String sFrontTiffFileName, String sBackTiffFileName,

String sFrontJPEGFileName, String sBackJPEGFileName, String[] asMICR, int[] aiFinalImageQuality, int[] aiFinalContrast, int[] aiDocStatus);

This function takes all the best parts of the DCCAPI to return the best image and MICR data. It uses Edge Detection, re-thresholding, Image Quality Analysis, MICR OCR and MICR Verification. This function was defined to simplify developer interface.

DccScanToFile - Scan one check and place the images in the files named by the passed parameters.

PARAMETERS:

sFrontTiffFileName - Front Image File Name or NULL

sBackTiffFileName - Back Image File Name or NULL

sFrontJPEGFileName- Front Image File Name or NULL

sBackJPEGFileName - Back Image File Name or NULL

asMICR - MICR Code Buffer (96 bytes)

aiFinalImageQuality - Final Weighted Image Quality if Front Tiff Requested

-1 if test not completed

0 no defects were found

1 to 100 weighted number of speckles found

Above 100 significant image defects found depending on test preformed.

aiFinalContrast -200 to 1000 if EdgeDetection or 2 to 13 if Simple

Thresholding

aiDocStatus - Copy of IQA Status Data if not a NULL Pointer. See IQAStatus for more

information. Must be allocated to 32 Integers (32 \* 4 bytes) if used.

Other Variables (existing):

JPEG Quality CFG\_MISC\_JPEG\_QUALITY

NonEdgeDetectionFrontThreshold CFG\_IMAGE\_FRONT\_BW\_THRESH

Magnetic MICR Errors CFG\_MISC\_MICR\_VERIFY

4 bit, 8 bit, or 24 bit capture CFG\_IMAGE\_GRAY256LEVEL,

CFG\_IMAGE\_FRONTCOLOR

Other Variables (new):

CFG\_DCCSCAN\_IQATESTS 146 //See IQAStatus Function Call

CFG\_DCCSCAN\_STARTCONTRAST 147 //Usually around 450

CFG\_DCCSCAN\_ENDCONTRAST 148 //Usually 450 to 900

CFG\_DCCSCAN\_INCCONTRAST 149 //Usually 100 to 600

CFG\_JPEG\_IMAGE\_RESOLUTION 150 //100 or 120 or 200 dpi

CFG\_DCCSCAN\_MAXSPECKLES 151 //Application Depend, Check 1200 to 2000

CFG\_DCCSCAN\_MINMICRQUALITY 152 //MICR Quality 5 to 10

CFG\_DCCSCAN\_MAXBENTCORNER 153 //Number of pixels in a bent corner 60 to 200

CFG\_DCCSCAN\_MAXRAWDENSITY154 //Usually 26 and up, but all business checks could be lower

CFG\_DCCSCAN\_MAXTIFFIMAGESIZE 155 //Maximum allowable Tiff Image Size in Kilobytes.

CFG\_DCCSCAN\_OPTIONS 156 //1 - try multiple EdgeDetection Contrast

//2 - try simple thresholding

//4 - despeckle the images

//8 – save grayscale and color images as BMP

// instead of JPEG

//16 - Group 4 Filter – Smaller Tiff Images

//32 - Return Floyd Steinberg Tiff as Grayscale

//64 - Force EdgeDetect Images to 3.1% Density

//128- Return grayscale/color as uncompressed

// image with a BUICMemHdr Header

//256-Test the Rear Tiff B/W image for density //and return the value in iDocStatus[31]

//512-Reduce 300 dpi to 240 dpi on CX30

//1024-Test Upside down documents, Optical

// MICR ONLY if found.

//2048-On CX30 Only, Process MICR forward and

//Backward to attempt to read very low

//intensity MICR

//4096-Reduce a 300 dpi image to 200 dpi. 300

//scanning is needed by Cmc7 OCR, but most

// applications want 200 dpi images.

//8192-Use a simple threshold instead of Edge

//Detection threshold on the back Tiff Image.

//16384 - Check Reversed Images –

// SB500/SB600 Only

//32768 – Reduce 600 to 200 DPI Front – SB600

//65536-Avoid "VOIDS" on Business Checks

//131072-Second Pass MICR OCR

//262144-Scale 600 dpi to 200 dpi front/back

CFG\_DCCSCAN\_MINLENGTH 192 // Minimum Document Length in tenth of inches.

// Default: 30

CFG\_DCCSCAN\_FIX\_IMAGES 212 //If 0 (default) disabled, 1 to 50 is number of K

// forces G4 Filter which can reduce G4 image

// Size by 10 to 60%

Examples

// Capture 3 images, Front Tiff, Back Tiff and Front JPEG

String sFrontTiffFileName = "C:\\Temp\\TestFront.tif";

String sBackTiffFileName = "C:\\Temp\\TestBack.tif";

String sFrontJPEGFileName = "C:\\Temp\\TestFront.JPG";

String sBackJPEGFileName = null;

String[] asMICR = {"0"};

int[] aiFinalImageQuality = {0};

int[] aiFinalContrast = {0};

int[] aiDocStatus = new int[32];

DccScanApi scanApi = new DccScanApi();

scanApi.DccScanToFile(sFrontTiffFileName,sBackTiffFileName,

sFrontJPEGFileName,sBackJPEGFileName,

asMICR , aiFinalImageQuality, aiFinalContrast, aiDocStatus);

## DccScanToMemory()

int DccScanToMemory(byte[] abFrontTiffImage, int[] aiFrontTiffImageLength,

byte[] abBackTiffImage, int[] aiBackTiffImageLength, byte[] abFrontJPEGImage, int[] aiFrontJPEGImageLength, byte[] abBackJPEGImage, int[] aiBackJPEGImageLength, String[] asMICR, int[] aiFinalImageQuality, int[] aiFinalContrast, int[] aiDocStatus);

This function takes all the best parts of the DCCAPI to return the best image and MICR data. It uses Edge Detection, re-thresholding, Image Quality Analysis, MICR OCR and MICR Verification. This function was defined to simplify developer interface.

DCCScan - Scan one check and place the images in the buffers named by the passed parameters.

PARAMETERS:

abFrontTiffImage - Copy of Front TIFF Image Data if not NULL. Must be allocated

to 512000 bytes if used.

aiFrontTiffImageLength - Copy of Front TIFF Image Length in bytes

abBackTiffImage - Copy of Back TIFF Image Data if not NULL. Must be allocated

to 512000 bytes if used.

aiBackTiffImageLength - Copy of Back TIFF Image Length in bytes

abFrontJPEGImage - Copy of Front JPEG Image Data if not NULL. Must be

allocated to 512000 bytes if used.

aiFrontJPEGImageLength - Copy of Front JPEG Image Length in bytes

abBackJPEGImage - Copy of Back JPEG Image Data if not NULL. Must be

allocated to 512000 bytes if used.

aiBackJPEGImageLength - Copy of Back JPEG Image Length in bytes

asMICR - MICR Code Buffer (96 bytes)

aiFinalImageQuality - Final Weighted Image Quality if Front Tiff Requested

-1 if test not completed

0 no defects were found

1 to 100 weighted number of speckles found

Above 100 significant image defects found depending on test preformed.

aiFinalContrast -200 to 1000 if EdgeDetection or 2 to 13 if

implement Thresholding

aiDocStatus - Copy of IQA Status Data if not a NULL Pointer. See IQAStatus for

more information. Must be allocated to 32 Integers (32 \* 4 bytes) if

used.

Other Variables (existing):

JPEG Quality CFG\_MISC\_JPEG\_QUALITY

NonEdgeDetectionFrontThreshold CFG\_IMAGE\_FRONT\_BW\_THRESH

Magnetic MICR Errors CFG\_MISC\_MICR\_VERIFY

4 bit, 8 bit, or 24 bit capture CFG\_IMAGE\_GRAY256LEVEL,

CFG\_IMAGE\_FRONTCOLOR

Other Variables (new):

CFG\_DCCSCAN\_IQATESTS 146 //See IQAStatus Function Call

CFG\_DCCSCAN\_STARTCONTRAST 147 //Usually around 450

CFG\_DCCSCAN\_ENDCONTRAST 148 //Usually 450 to 900

CFG\_DCCSCAN\_INCCONTRAST 149 //Usually 100 to 600

CFG\_JPEG\_IMAGE\_RESOLUTION 150 //100 or 120 or 200 dpi

CFG\_DCCSCAN\_MAXSPECKLES 151 //Application Depend, Check 1200 to 2000

CFG\_DCCSCAN\_MINMICRQUALITY 152 //MICR Quality 5 to 10

CFG\_DCCSCAN\_MAXBENTCORNER 153 //Number of pixels in a bent corner 60 to 200

CFG\_DCCSCAN\_MAXRAWDENSITY154 //Usually 26 and up, but all business checks could be lower

CFG\_DCCSCAN\_MAXTIFFIMAGESIZE 155 //Maximum allowable Tiff Image Size in Kilobytes.

CFG\_DCCSCAN\_OPTIONS 156 //1 - try multiple EdgeDetection Contrast

//2 - try simple thresholding

//4 - despeckle the images

//8 – save grayscale and color images as BMP

// instead of JPEG

//16 - Group 4 Filter – Smaller Tiff Images

//32 - Return Floyd Steinberg Tiff as Grayscale

//64 - Force EdgeDetect Images to 3.1% Density

//128- Return grayscale/color as uncompressed

// image with a BUICMemHdr Header

//256-Test the Rear Tiff B/W image for density //and return the value in iDocStatus[31]

//512-Reduce 300 dpi to 240 dpi on CX30

//1024-Test Upside down documents, Optical

// MICR ONLY if found.

//2048-On CX30 Only, Process MICR forward and

//Backward to attempt to read very low

//intensity MICR

//4096-Reduce a 300 dpi image to 200 dpi. 300

//scanning is needed by Cmc7 OCR, but most

// applications want 200 dpi images.

//8192-Use a simple threshold instead of Edge

//Detection threshold on the back Tiff Image.

//16384 - Check Reversed Images –

// SB500/SB600 Only

//32768 – Reduce 600 to 200 DPI Front – SB600

//65536-Avoid "VOIDS" on Business Checks

//131072-Second Pass MICR OCR

//262144-Scale 600 dpi to 200 dpi front/back

CFG\_DCCSCAN\_MINLENGTH 192 // Minimum Document Length in tenth of inches.

// Default: 30

CFG\_DCCSCAN\_FIX\_IMAGES 212 //If 0 (default) disabled, 1 to 50 is number of K

// forces G4 Filter which can reduce G4 image

// Size by 10 to 60%

Examples

// Capture 3 images, Front Tiff, Back Tiff and Back JPEG

byte[] abFrontTiffImageBuffer = new byte[1024\*512];

byte[] abBackTiffImageBuffer = new byte[1024\*512];

byte[] abFrontJPEGImageBuffer = null;

byte[] abBackJPEGImageBuffer = new byte[1024\*512];

int[] aiFrontTiffImageLength = {0};

int[] aiBackTiffImageLength = {0};

int[] aiFrontJPEGImageLength = null;

int[] aiBackJPEGImageLength = {0};

String[] asMICR = {"0"};

int[] aiFinalImageQuality\_1 = {0};

int[] aiFinalContrast = {0};

int[] aiDocStatus\_1 = new int[32];

DccScanApi scanApi = new DccScanApi();

scanApi.DccScanToMemory(abFrontTiffImageBuffer, aiFrontTiffImageLength,

abBackTiffImageBuffer,aiBackTiffImageLength,

abFrontJPEGImageBuffer,aiFrontJPEGImageLength,

abBackJPEGImageBuffer,aiBackJPEGImageLength,

asMICR,aiFinalImageQuality,aiFinalContrast,

aiDocStatus);

## DccScanSetSpecialDocument()

int DccScanSetSpecialDocument(int iCount, int iIndex, String sRouting,

String sAccount, int iThreshold, int iMin, int iContrast, int iOptions)

This function allows the user to override DccScanToFile and DccScanToMemory. It handles documents based on their routing and account numbers in a special way. This function must be called once for every supported document type.

PARAMETERS:

int iCount Total number of document types for special handling. Changing this number will clear previous settings. Valid from 0 to 100.

int iIndex 1 to iCount,

String sRouting Routing or Bank number, usually :123456789: in U.S.A.

String sAccount Optional Customer Account Number 123457 or blank string (“”)

int iThreshold 1 - Edge Detection Thresholding

2 - Simple Thresholding

3 - Floyd Steinberg Threshold

int iMin Minimum Background white Pixels Required. Valid from 10 to 80 and this requires the percentage of pixels to be treated as white background. This can be

increased to sometimes skip complex document backgrounds that make edge detection thresholding too noisy.

int iContrast 200 to 1024 when using threshold 1 (Edge Detection Algorithm),

2 to 13 when using threshold 2 (Simple Thresholding), and

Ignored when threshold 3 (Floyd Steinberg Threshold).

int iOptions 0-Off

1 - Group 4 Filter to get smaller Tiff Images

2 - Ignore Speckles when calculating Image Quality

4 - Remove Speckles

8 - Ignore Bent Corners when calculating Image Quality.

Some images have dark corners that always look bent.

Options can be combined (added), so 7 would turn on the first three options.

16-Ignore Streaks – Some documents have multiple horizontal lines that look like streaks.

32 – Ignore Double Feed Sensor, since this document type will always force a DF Error

64- Ignore Compression Size/Density IQA Test

128 – If Double Feed Sensor Triggered, then force a DF Error.

256 – If OCR MICR is good, Magnetic is questionable, then override MICR

RETURN:

0 - Okay

BUIC\_ERROR\_BADPARAM

BUIC\_ERROR\_MEMORY\_ALLOC

## DccScanSetSpecialDocumentEx()

int DccScanSetSpecialDocumentEx(int iCount,

int iIndex, String sRouting, String sAccount, int iThreshold, int iMin,

int iContrast, int iOptions, int iSecondThreshold, int iSecondContrast,

int iLeft, int iTop, int iRight, int iBottom);

This function allows the user to override DCCScan and handle documents based on their routing and account numbers in a special way. This function must be called once for every supported document type. Special Documents is the method users can use to override image thresholding, double feed detection, and support non-magnetic MICR checks on a customer account bases.

PARAMETERS:

int iCount Total number of document types for special handling. Changing this number will clear previous settings. Valid from 0 to 100.

int iIndex 1 to iCount,

String sRouting Routing or Bank number, usually :123456789: in U.S.A.

String sAccount Optional Customer Account Number 123457 or blank string (“”)

int iThreshold 1 - Edge Detection Thresholding

2 - Simple Thresholding

3 - Floyd Steinberg Threshold

int iMin Minimum Background white Pixels Required. Valid from 0 to 80 and this requires the percentage of pixels to be treated as white background. This can be

increased to sometimes skip complex document backgrounds that make edge detection thresholding too noisy. Usually set to zero.

int iContrast 200 to 1024 when using threshold 1 (Edge Detection Algorithm),

2 to 13 when using threshold 2 (Simple Thresholding), and

Ignored when threshold 3 (Floyd Steinberg Threshold).

int iOptions 0-Off

1 - Group 4 Filter to get smaller Tiff Images

2 - Ignore Speckles when calculating Image Quality

4 - Remove Speckles

8 - Ignore Bent Corners when calculating Image Quality.

Some images have dark corners that always look bent.

Options can be combined (added), so 7 would turn on the first three options.

16-Ignore Streaks – Some documents have multiple horizontal lines that look like streaks.

32 – Ignore Double Feed Sensor, since this document type will always force a DF Error

64- Ignore Compression Size/Density IQA Test

128 – If Double Feed Sensor Triggered, then force a DF Error.

256 – If OCR MICR is good, Magnetic is questionable, then override MICR

int iSecondThreshold

0 – No Second Threshold

1 - Edge Detection Thresholding

2 - Simple Thresholding

int iSecondContrast

200 to 1024 when using threshold 1 (Edge Detection Algorithm),

2 to 13 when using threshold 2 (Simple Thresholding),

Int iLeft Left Edge in 10th of inches

Int iTop Top Edge in 10th of inches

Int iRight Right Edge in 10th of inches

Int iBottom Bottom Edge in 10th of inches

RETURN:

0 - Okay

BUIC\_ERROR\_BADPARAM

BUIC\_ERROR\_MEMORY\_ALLOC

## DccScanGetOcrMicr()

int DccScanGetOcrMicr(String[] asMICR, String[] asMICRConf)

This function allows the user to get the OCR MICR results after DCCScan () is called. These results are OCR (Opitical Characters Recognition) only and are usually only useful if the scanned documents was not printed with magnetic ink or the MICR line was too high to read. DCCScan() always uses the magnetic MICR results first and OCR MICR to improve or correct magnetic results. Documents without magnetic ink are more likely frauds. CMC7 OCR only works on 300 dpi images.

asMICR Actual E13B/CMC7 MICR results in character form. Buffer should be preallocated to accept upto 128 characters.

asMICRConf A characters string were each character (0 to 4) corresponds to the confidence where 0,1, are low confidence, 2 is okay, and 3 and 4 are good E13B/CMC7 character confidence.

## DccScanLong()

int DccScanLong(String sFrontFileName, byte[] abFrontImage, int[] aiFrontFileSize, int iFormatComp);

DccScanLong will scan a paper tape 4 inches tall by up to 144 inches on a TS240.

Parameters:

|  |  |
| --- | --- |
| sFrontFileName | Full Path to file to store compressed image |
|  |  |
|  |  |
| abFrontImage | Buffer for uncompressed grayscale image if sFrontFileName is not specified or NULL. |
|  |  |
| aiFrontFileSize | Compressed image Size |
|  |  |
| iImageFormat | Select image file format as either -bit TIFF or single component (gray scale) JPEG. Use FORMAT\_TIFF or FORMAT\_JPEGG only. |

## BuicScanGray()

int BuicScanGray(int iJobType, String sFront, String[] asLenFront, String sBack, String[] asLenBack, String[] asCode, String[] asLenCode, int iImageFormat);

BuicScanGray - Scan one check in gray scale mode and place the image in the files named by the passed parameters. iJobType specifies which images (information) is captured and stored.

PARAMETERS:

iJobType -

1 - Front

2 - Back

3 - Front and Back

4 - MICR

5 - Front and MICR

6 - Back and MICR

7 - Front, Back and Micr

sFront - File name for front of the check image.

asLenFront - Length of the file (ASCII) for the front of the check image.

sBack - File name for back of the check image.

asLenBack - Length of the file (ASCII) for the back of the check image.

asCode - MICR buffer of the check.

asLenCode - Length of the file (ASCII) containing the MICR.

iImageFormat - Select image format for front/back file (BMP, JPEG Gray)

Use FORMAT\_BMP and FORMAT\_JPEGG.

RETURN:

TRUE - Everything okay.

BUIC\_ERROR\_NOINIT - Have not initialized the scanner

before starting this scan.

BUIC\_ERROR\_BADJOB - Job type not within range of known types.

BUIC\_GRAYFILE - Gray scale not supported.

## BuicScanMemoryGray()

int BuicScanMemoryGray(int iJobType, byte[] abFront, int[] aiLenFront, byte[] abBack, int[] aiLenBack, String[] asCode, int[] aiLenCode, int ImageFormat);

BuicScanMemoryGray - Scan one check and place the image in the memory specified by the passed parameters. iJobType specifies which images (information) is captured and stored.

PARAMETERS:

iJobType -

1 - Front

2 - Back

3 - Front and Back

4 - MICR

5 - Front and MICR

6 - Back and MICR

7 - Front, Back and Micr

abFront - Buffer for front of the check image.

aiLenFront - Length for the front of the check image.

abBack - Buffer for back of the check image.

aiLenBack - Length for the back of the check image.

sCode - MICR string of the check.

aiLenCode - Length the MICR.

iImageFormat - Select image format for front/back file (BMP, JPEG Gray)

Use FORMAT\_BMP and FORMAT\_JPEGG.

RETURN:

TRUE - Everything okay.

BUIC\_ERROR\_NOINIT - Have not initialized the scanner

before starting this scan.

BUIC\_ERROR\_BADJOB - Job type not within range of known types.

## BuicScan()

int BuicScan(int iJobType, String sFront, String[] asLenFront, String sBack, String[] asLenBack, String[] asCode, String[] asLenCode);

BuicScan - Scan one check and place the imagein the files named by the passed parameters. iJobType specifies which images (information) is captured and stored.

PARAMETERS:

iJobType -

1 - Front

2 - Back

3 - Front and Back

4 - MICR

5 - Front and MICR

6 - Back and MICR

7 - Front, Back and Micr

sFront - File name for front of the check image.

asLenFront - Length of the file (ASCII) for the front of the check image.

sBack - File name for back of the check image.

asLenBack - Length of the file (ASCII) for the back of the check image.

asCode - MICR buffer of the check.

asLenCode - Length of the file (ASCII) containing the MICR.

RETURN:

TRUE - Everything okay.

BUIC\_ERROR\_NOINIT - Have not initialized the scanner

before starting this scan.

BUIC\_ERROR\_BADJOB - Job type not within range of known types.

## BuicScanMemory()

int BuicScanMemory(int iJobType, byte[] abFront, int[] aiLenFront, byte[] abBack, int[] aiLenBack, String[] asCode, int[] aiLenCode);

BuicScanMemory - Scan one check and place the image in the memory specified by the passed parameters. iJobType specifies which images (information)is captured and stored.

PARAMETERS:

iJobType -

1 - Front

2 - Back

3 - Front and Back

4 - MICR

5 - Front and MICR

6 - Back and MICR

7 - Front, Back and Micr

abFront - Buffer for front of the check image.

aiLenFront - Length of the front of the check image.

abBack - Buffer for back of the check image.

aiLenBack - Length of the back of the check image.

sCode - MICR string of the check.

aiLenCode - Length of the MICR.

RETURN:

TRUE - Everything okay.

BUIC\_ERROR\_NOINIT - Have not initialized the scanner

before starting this scan.

BUIC\_ERROR\_BADJOB - Job type not within range of known types.

## DccScanUV()

int DccScanUV(String sFrontFileName, byte[] abFrontImage, int[] aiFrontFileSize, int iFormatComp);

DCCScanUV will return the UV image with the CFG\_IMAGE\_FRONTCOLOR is set to CFG\_IMAGE\_DUALGRAYUV. Function must be called only after a successful DCCScan(…).

Parameters:

|  |  |
| --- | --- |
| sFrontFileName | Full Path to file to store compressed image |
|  |  |
|  |  |
| abFrontImage | Buffer for uncompressed grayscale image if sFrontFileName is not specified or NULL. |
|  |  |
| aiFrontFileSize | Compressed image Size |
|  |  |
| iImageFormat | Select image file format as either -bit TIFF or single component (gray scale) JPEG. Use FORMAT\_TIFF, FORMAT\_TIFF\_REVERSE, or FORMAT\_JPEGG only. |

# Endorsement API Functions

The TS215, TS220, TS230, and TS240 measure the offset of endorsement start position from the right hand side of the rear document. An offset on these scanners move the endorsement text to the left if viewing the rear of a document. Offset are normally 200 dots per inch.

The CX30 and TS4120 measure the offset of the endorsement start position from the left hand side of the rear document.

For scanners that can print multiple lines of text the carriage return is used to indicate the start of the next line. For example the following string would print on four different lines. “Line 1\nLine 2\nLine 3\nLine 4”.

## DccBatchPrintBMP()

int DccBatchPrintBMP(byte[] abPrintString, long lStartPosition, long lMode,

int iStartCounter, int iIncrement, int iTextHeight, int iXOffset, int iYOffset, int iBold, int iItalic, String sFont);

This function is used on scanners that allow Batch Processing or the ability to start feeding the next document while processing the current document. The string will have a print string with a counter that the low level modules will automatically update and send to the printer unit. Because documents are being fed ahead of time and past the printer unit before the document is acknowledged by the application, print strings cannot be set by the application in batch mode.

e.g. DccBathPrintBMP(“Decrement %7d by 5’s”, 800, 0, 100, -5, 16, 0, 1, 1, 0, “Arial”); would have the following printed on the back of the documents:

First Document:: Decrement 100 by 5’s

Second Document: Decrement 95 by 5’s

Third Document Decrement 90 by 5’s

Parameters:

|  |  |
| --- | --- |
| abPrintString | String to print with a %d or %5d in it for the counter |
| lStartPosition | Position to start printing |
| lMode | 0 – default  1 – CLEAR\_PREVIOUS  2 – INVERT\_COLOR  256 – Update String across Physical Batches of documents |
| iStartCounter | Value to be place in counter for first document |
| iIncrement | Value to change counter for each subsequent documents. Note negative numbers mean decrementing. |
| iTextHeight | Height in Pixels. |
| iXOffset | String offset in pixels – Space before first character in the string. |
| iYOffset | Offset in pixels to move text up/down. |
| iBold | Bold characters or Not. TRUE (1)/FALSE (0). |
| iItalic | Italic characters or Not. TRUE (1)/FALSE (0). |
| sFont | Pointer to name of a FONT. “Arial”, “Haettenschweiler” or other True Type Font. |

Notes: **CFG\_MISC\_SCANBATCH\_ENABLE must be on. Otherwise the function will not work as expected.**

## DccBatchPrintString()

int DccBatchPrintString(byte[] abPrintString, long lStartPosition, long lMode, int iStartCounter, int iIncrement)

This function is used on scanners that allow Batch Processing or the ability to start feeding the next document while processing the current document. The string will have a print string with a counter that the low level modules will automatically update and send to the printer unit. Because documents are being fed ahead of time and past the printer unit before the document is acknowledged by the application, print strings cannot be set by the application in batch mode.

e.g. DccBathPrintString(“Decrement %7d by 5’s”, 800, 0, 100, -5); would have the following printed on the back of the documents:

First Document:: Decrement 100 by 5’s

Second Document: Decrement 95 by 5’s

Third Document Decrement 90 by 5’s

Parameters:

|  |  |
| --- | --- |
| abPrintString | String to print with a %d or %5d in it for the counter |
| lStartPosition | Position to start printing |
| lMode | 0 – default  1 – CLEAR\_PREVIOUS  2 – INVERT\_COLOR  256 – Update String across Physical Batches of documents |
| iStartCounter | Value to be place in counter for first document |
| iIncrement | Value to change counter for each subsequent documents. Note negative numbers mean decrementing. |

Notes: **CFG\_MISC\_SCANBATCH\_ENABLE must be on. Otherwise the function will not work as expected.**

## DccScanVirtualEndorsement ()

int DccScanVirtualEndorsement(byte[] abVirtualString, long lStartPosition,

long lMode, int iStartCounter, int iIncrement, int iTextHeight, int iLeft,

int iTop, int iBold, int iItalic, int iRotate, int iDitherMode, int iRed, int iGreenGray, int iBlue, String sFont);

This function adds a virtual endorsement text onto images scanned in using DccScanToFile or DccScanToMemory. This function can be used to create virtual endorsements on images when using DccScanToFile or DccScanToMemory.

Parameters:

|  |  |
| --- | --- |
| byte[]abVirtualString | Text information to be added to image. Text information may include  one %d string and may include line feeds. The %d value is filled with iStartCounter for the first image and each subsequent image is  iStartCounter + iIncrement \* ImageNumber  Front, back, or both images selectable. |
| long lStartPosition | NOT USED |
| long lMode | 0 --- Disable Front and Back Endorsement  1 --- Front side of image  2 --- Back side of image  3 --- Front and Back side of image  NOTE: 3 is used for setting the same string to both front and back, use separate calls for different strings to front and back |
| int iStartCounter | Value for first document %d value or insertion |
| int iIncrement | Value to increment iStartCounter for each subsequent document |
| int iTextHeight | Text Height in Pixels. |
| int iLeft | Starting position in pixels from left edge of check. To use the right edge as a reference, multiple the position by -1. |
| int iTop | Starting position in pixels from top edge of check. |
| int iBold | Bold characters or Not. TRUE (1)/FALSE (0). |
| int iItalic | Italic characters or Not. TRUE (1)/FALSE (0). |
| int iRotate | 1 – rotate 90 degrees, 2 – rotate 180 degrees, 3 – rotate 270 degrees, 0 – no action |
| int iDitherMode |  |
| int iRed | Shade of red for color image |
| int iGreenGray | Shade of green for color image or shade of gray for gray image |
| int iBlue | Shade of blue for color image |
| String sFont | Pointer to name of a FONT. “Arial”, “Haettenschweiler” or other True Type Font. |

# Other API Functions

## BuicStatus()

int BuicStatus()

This function queries the scanner to see whether or not a check is in the feeder. Returns 0 if no check is found, 1 if a check is found, or greater for successful completion and a General Error code if an error occurs.

## BuicStatusDelay()

int BuicStatusDelay(int iIntervalMilliSec, int iTotalDelayMilliSec)

This function queries the scanner to see whether or not a check is in the feeder and will intelligently keep checking if a check is not present. Returns 0 if no check is found after timeout, 1 if a check is found, or a General Error code if an error occurs.

|  |  |
| --- | --- |
| iIntervalMilliSec | Delay between calls to BuicStatus if feeder is empty. Value Range 16 to 500 (half a second). |
| iTotalDelayMilliSec | If this time is exceeded, then just return empty feeder (0).  1 Second would be 1000  1 Minute would be 60000  1 Hour would be 60 \* 60 \* 1000 or 3600000  1 Day would be 24 \* 60 \* 60 \* 1000 or 86400000 |

Returns:

|  |  |
| --- | --- |
| 1 | Check Present |
| 0 | Feeder Empty and timeout |
| Other | General Error Code |

## DccApiVersion()

int DccApiVersion()

DccApiIVersion returns the version number of the API DLL being called by the application. The version is returned as a whole number. Example: Version 5.20 would be returned as 520.

## DccApiSupportedScanners()

int DccApiSupportedScanners()

DccApiSupportedScanners returns a long integer containing the devices supported by the API DLL being called. A bit-wise comparison may be used to determine which scanners are supported.

Returns:

If the LSB (Bit 0) is set, BUIC 1000 supported. (0x0001)

If Bit 1 (Bit 1) is set, BUIC 1500 supported. (0x0002)

If Bit 2 (Bit 2) is set, TS400 supported. (0x0004)

If Bit 3 (Bit 3) is set, TS200 supported. (0x0008)

If Bit 4 (Bit 4) is set, TS300 supported. (0x0010)

If Bit 5 (Bit 5) is set, TS300EBS supported. (0x0020)

If Bit 6 (Bit 6) is set, TS220 supported. (0x0040)

If Bit 7 (Bit 7) is set, TS230-65 and TS230-100 supported. (0x0080)

If Bit 8 (Bit 8) is set, TS4120 supported. (0x0100)

If Bit 9 (Bit 9) is set, CX30 supported. (0x0200)

If Bit 10 is set, TS240 supported. (0x0400)

If Bit 11 is set, SB500, SB600, SB650 are supported. (0x0800)

If Bit 12 is set, BX7200 is supported. (0x1000)

**NOTE:** *Values returned from DccApiSupportedScanners() do not equate to the values used for GetScannerType.*

## GetScannerType()

int GetScannerType()

This function returns the type of the currently detected scanner.

If you really want the scanner type please use BuicGetParam(CFG\_MISC\_SCANNER\_TYPE);

## BuicGetScannerInfo()

int BuicGetScannerInfo(int[] aiAdapterID, int[] aiTargetID, String[] asVendor,

String[] asProduct, String[] asRelease, int iMessageBox);

This function issues a SCSI inquiry to the adapter ID and target ID address, and returns the vendor, product, and release information. If a scanner is not found at this address, BuicGetScannerInfo starts searching all valid SCSI addresses for a scanner.

Parameters:

|  |  |
| --- | --- |
| aiAdapterID | Pointer to adapter ID. SCSI ID numbers should range from 0 to 7 and numbering starts at 0. If BuicGetScannerInfo will return the adapter ID as is unless it has to search for the scanner and then it will return the Adapter ID of the scanner found. |
| aiTargetID | Pointer to target ID. SCSI ID numbers should range from 0 to 7. BuicGetScannerInfo will return the target ID as is unless it has to search for the scanner and then it will return the Target ID of the scanner found. |
| asVendor | Vendor information returned from SCSI inquiry. Buffer must be at least 32 bytes long. |
| asProduct | Product information returned from SCSI inquiry. Buffer must be at least 32 bytes long. |
| asRelease | Release information returned from SCSI inquiry. Buffer must be at least 32 bytes long. |
| iMessageBox | If iMessageBox is set to 1, a message box with the vendor, product, and release information will be displayed. |

## BuicGetScannerSerialNumber()

int BuicGetScannerSerialNumber(String[] asSerialNumber, int[]

aiDocumentNumber, int[] aiTime)

Parameters:

|  |  |
| --- | --- |
| asSerialNumber | Pointer to serial number. |
| aiDocumentNumber | Pointer to number of documents to the nearest 10th document.  i.e. at every tenth document the counter is updated by 10. |
| aiTime | Pointer to time in Minutes.  i.e. to nearest 10 minutes. |

**NOTE:** *SCANNER UPDATES NUMBER of DOCUMENTS and TIME in MINUTES at random intervals. It may be every minute, but possibly even a longer or shorter interval. Therefore, number of documents and minutes is only an approximation since it is updated randomly and is only to the nearest 10.*

Returns:

0 or greater for successful completion or a General Error code if an error occurs.

## BuicCropFile()

short BuicCropFile(String sInputFile, String sOutputFile,

short Margin)

This function crops a TIFF image by removing white border around the image and then stores the newly created image in a new file. Return is 0 or greater for successful completion. A Window Error Code or a General Error Code is returned when an error occurs.

Parameters:

|  |  |
| --- | --- |
| sInputFile | An ASCII string containing the file specification including full drive, path, name, and extension of the image to be cropped. |
| sOutputFile | An ASCII string containing the file specification including full drive, path, name, and extension of the image to be created. |
| Margin | A Percentage (0-5) of border to add to the cropped image. If 0 is specified, then all white borders will be removed. If 5 is specified, after cropping all white borders, a small white border will be added equal to 5% of the total image size to enhance displays in certain applications. If -1, no crop is done, but the width is forced to a multiple of 8. |

**Note:** This function is unnecessary for TellerScan™ users.

## BuicCombineTIFFS()

short BuicCombineTIFFS(String sInputFile, String sAppendFile)

BuicCombineTIFFS appends a TIFF image to a previously created TIFF image file to store multiple images in one file. Return is 0 or greater for successful completion. A Window Error Code or a General Error Code is returned when an error occurs.

Parameters:

|  |  |
| --- | --- |
| sInputFile | An ASCII string containing the file specification including full drive, path, name, and extension of the multi-page TIFF image file. |
| sAppendFile | An ASCII string containing the file specification including full drive, path, name, and extension of the image to be appended to InputFile. |

## BuicDebug()

int BuicDebug(int iState)

BuicDebug - function writes out messages. Written debug message if iState is TRUE. This method of debug does not really work, but is left for backward compatibility.

PARAMETERS:

iState 0 - BUIC\_DEBUG\_OFF

1- BUIC\_DEBUG\_ON

RETURN:

TRUE - Everything initialized correctly.

## BuicCopyFile()

short BuicCopyFile(String sInputFile, String sOutputFile)

This function copies a file.

PARAMETERS:

sInputFile - Source file name.

sOutputFile - Destination file name.

Return: 0 OKAY

Negative - General Error

## BuicCompressImage()

int BuicCompressImage(String sFileName, byte[] abhugeImage);

BuicCompressImage - Compress image.

PARAMETERS:

psFileName - Pointer to file name to store image.

pchugeImage - Pointer to image stored in memory to be compressed.

RETURN:

TRUE - Everything okay.

BUIC\_ERROR\_IMAGE\_TYPE - Only support 1 bit per pixel at this time.

BUIC\_ERROR\_IMAGE\_TYPE - Image must be have dimensions greater than 0

## BuicCompressImageGray()

int BuicCompressImageGray(String sFileName, byte[] abhugeImage, int iImageFormat);

BuicCompressImageGray - Compress image grayscale.

PARAMETERS:

sFileName - Pointer to file name to store image.

abhugeImage - Pointer to image stored in memory to be compressed.

iImageFormat - Select image format for front/back file (BMP, JPEG Gray)

Use FORMAT\_BMP and FORMAT\_JPEGG.

RETURN:

TRUE - Everything okay.

BUIC\_ERROR\_IMAGE\_TYPE - Only support 1 bit per pixel at this time.

BUIC\_ERROR\_IMAGE\_TYPE - Image must be have dimensions greater than 0

## DccUVGain()

int DccUVGain(String sInputImage, String sOutputImage, int iGainFactor, int iBlackLevel, int iGainPlusMinus);

DccUVGain - Change Gain of UV Grayscale Image and optionally set

BlackLevel.

PARAMETERS:

sInputImage - Input image file name.

sOutputImage - Output image file name.

iGainFactor - Gain factor

iBlackLevel - Black level

iGainPlusMinus – Gain increment/decrement

RETURN:

1. - Everything okay.

negative value - DCC Error Code

## DccUVTiff ()

int DccUVTiff(String sInputImage, String sOutputImage, int iThreshold);

DccUVTiff - Convert UV Grayscale Image to TIFF

PARAMETERS:

sInputImage - Input image file name.

sOutputImage - Output image file name.

iThreshold – Simple threshold value

RETURN:

0 - Everything okay.

negative value - DCC Error Code