

# Pixel Translations ISIS Scanning ActiveX Control

---

**User's Guide  
ImageBASIC 3.1**



Diamond Head Software, Inc.  
1217 Digital Drive Ste. 125  
Richardson, Texas 75081  
(972) 479-9205

## **COPYRIGHT NOTICES**

Under the copyright laws, neither the documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of Diamond Head Software, Inc., except in the manner described in the documentation.

This software product contains proprietary software components developed by a number of different software companies, referred herein as "Third Party Licensors". This documentation and the software that you purchased are protected by one or more of the following copyright notices:

Portions of this product, © 1994, 1995, 1996, 1997 Diamond Head Software, Inc. All rights reserved.  
Portions of this product, © 1993, 1994, 1995, 1996 Pixel Translations, Inc. All rights reserved.

Company and product names mentioned in this documentation are trademarks or registered trademarks of their respective companies. Lotus and Lotus Notes are registered trademarks of Lotus Development Corporation. Windows is a trademark and Microsoft is a registered trademark of Microsoft Corporation.

DIAMOND HEAD SOFTWARE INC. AND ITS THIRD PARTY LICENSORS MAKE NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE SOFTWARE. DIAMOND HEAD SOFTWARE, INC. AND ITS THIRD PARTY LICENSORS DO NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS OR OTHERWISE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY YOU. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME JURISDICTIONS. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU.

IN NO EVENT WILL DIAMOND HEAD SOFTWARE INC. OR ITS THIRD PARTY LICENSORS AND/OR THEIR DIRECTORS, OFFICERS, EMPLOYEES OR AGENTS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, AND THE LIKE) ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE EVEN IF DIAMOND HEAD SOFTWARE INC. OR ITS THIRD PARTY LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU. Diamond Head Software Inc.'s and its Third Party Licensors' liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort (including negligence), product liability or otherwise), will be limited to \$50.

# Contents

Chapter 1 : Introduction	1
Linking Controls.....	1
Licensing Configuration and Verification.....	2
Chapter 2 : Scanning	5
Introduction to Scanning.....	5
Programming Considerations.....	5
Scanner Hardware Setup.....	7
Scanner Selection.....	8
Scanner Selection -- Graphical.....	8
Scanner Selection -- Programmatic.....	9
Scanner Setup.....	11
Scanner Setup -- Graphical.....	11
Scanner Setup -- Programmatic.....	12
Saving and Loading a Scanner Setup.....	13
Selecting an Image Destination.....	15
Scanning to Another ImageBASIC Control.....	15
Scanning to File.....	15
Scanning One Page.....	17
Scanning a Batch.....	17
Scanning Events.....	20
Chapter 3 : Reference	21
Reference of Properties, Events and Methods.....	21
Appendix A : File Formats	43
File and Compression Formats.....	43
File Formats Supported by the PixScan Control.....	43
Compression Types.....	45
Color Limitations of File Formats.....	46
Appendix B : Scanner Control Tags	49
Overview of Tag Support.....	49
Scanner Tag Data Types.....	50
Ordering the Setting of Tags.....	54
Reference to the Scanner Control Tags.....	55
Resolution and Page Size Scanner Control Tags.....	55
Contrast and Brightness Scanner Control Tags.....	60

Gamma Correction.....	62
Color Definition Scanner Control Tags.....	63
Image Data Storage Scanner Control Tags.....	65
Image Data Interpretation Scanner Control Tags.....	69
Hardware and Software Descriptive Tags.....	72
Image Correction and Enhancement Scanner Control Tags.....	78
Endorser Support Scanner Control Tags.....	84
Image Quality Options through Scanner Control Tags.....	88
EEPROM Settings through Scanner Control Tags.....	89
Window and Subwindow Access Tags.....	90
Bar Code Recognition.....	93

## Index

101

# Chapter 1 : Introduction

---

## Linking Controls

Virtually all ImageBASIC controls can accept image data from other ImageBASIC controls. The process of designating where each ImageBASIC component gets its image data is referred to as linking the controls. With the exception of those controls that can directly access files or scanners, all ImageBASIC controls must be linked to another ImageBASIC control to get any image data.

Linking of controls is the primary method of moving an image through a series of processing steps. For example, an image may be originally captured through the PixScan ActiveX control, passed to a TMS Display control for operator verification, optionally routed through a ScanFix control for enhancement, then to a TMS File control to be written to disk , and finally to a TextBridge control for OCR processing to generate indexing information for that file.

### Creating the Link

Almost all of the ImageBASIC controls have a property named **ImageDataSource**. To create the data link between controls, this property must specify the ImageBASIC control that will be supplying image data. Any image that is received by the source control will also be sent to the linked control. For example, if the **ImageDataSource** property of a TextBridge control is set to a TMS Display control, each time a new image is loaded into the display window, the TextBridge control will receive that image.

### Linking at Design Time

As each ImageBASIC control is added to a Form at design time, its **ImageDataSource** property is automatically set to an ImageBASIC control already on that Form. The assignment may be changed at design time by selecting from the drop-down list of available ImageBASIC components. This list is shown with the **ImageDataSource** property in the Properties Window or Object Inspector.

### Linking at Runtime

Linking controls at runtime requires only one line of code that can be executed at any time. The source of image data can be changed during program execution by naming another ImageBASIC control in the **ImageDataSource** property, as shown here:

```
TMSDispl.ImageDataSource = PixScan1.Link
```

The **Link** property is automatically populated with a unique ID string each time a new instance of an ImageBASIC control is created. This ID string identifies which of the controls in a project supplies the image data. The **Link** property is used in all ImageBASIC control linking, but in Visual Basic **Link** is the default property and, therefore, does not need to be specified, as shown here:

```
TMSDispl.ImageDataSource = PixScan1
```

Each time the image data flowing from one ImageBASIC control changes, the receiving control's **ImageDataChanged** event is triggered. From this event, any procedure that is to be performed on each image can be started. For example, each time an OCR control's **ImageDataChanged** event occurs, an OCR attempt could be started on the new image data.

---

## Licensing Configuration and Verification

Licensing in ImageBASIC is based on enabling a set number of concurrent seats. The number of seats that can use ImageBASIC is controlled by access to licenses. A license is an entry in a database that allows one computer to run a specific ImageBASIC component.

For example, there is a special type of license for the TMS Display control, another for TextBridge, another for ScanFix, etc. Each one of these licenses also comes in two varieties, runtime and development.

As is suggested by the names, *runtime licenses* are necessary for an *executable* to function properly, and *development licenses* are necessary to *develop* an application using ImageBASIC. A design time license will also function as a runtime license, which obviates the need to add runtime licenses for testing applications during development.

### ***Where the Licenses are Kept***

ImageBASIC will find licenses stored in either one of two locations -- in a licensing database or on a hardware key. The hardware key is plugged into the parallel port of the computer using ImageBASIC and will be automatically found each time an ImageBASIC component is used. The licensing database must be created on the site where it will be used and may not be moved from the location in which it is installed.

The inability to move a licensing database is one of its protection features. This attachment to a single location is formed when the licensing database is first created. Although it may not be moved once created, the licensing database can be written to a network drive to which all the machines running ImageBASIC have access. A file called IMGBASIC.INI must be on each of these networked

machines. This file contains an entry pointing to the location of the licensing database. ImageBASIC can now search the licensing database for an available copy of each license it needs to run.

This licensing database may also be created on a local drive, activating ImageBASIC on only that one machine. The same INI file is still necessary to pinpoint the location of the database. If you opt to create a separate licensing database on each individual machine using ImageBASIC, it will of course contain only one copy of each type of license. This one copy is sufficient to activate any number of ImageBASIC-based applications on this one computer.

## ***How the Licenses are Used***

As each application that uses ImageBASIC is initiated, or the control is loaded into the development environment, the ImageBASIC licensing server attempts to find the proper license for each component. For example, if an executable has been developed that uses ImageBASIC to display existing images, and then calls on TextBridge to perform OCR on that image, that application must be able to find one available *TMS Display runtime license* and one available *TextBridge runtime license*.

If the application is successful in finding both these licenses, it will load normally and function exactly as it was programmed. When the application finds these licenses and is thereby informed that the correct licenses are available, it simultaneously locks the licenses so that no other application can use the same licenses to run at the same time. However, if two copies of these same licenses are available, another computer will be able to run the same application and will then lock the second license.

When an application that has locked one or more licenses terminates, the licenses are released and can immediately be taken by another user. This is the process by which concurrent licensing for any number of seats may be enabled. If the application ends abnormally -- the user might reboot or a concurrently running Windows application might lock up -- then the release of the licenses is conditional on the network/disk operating system.

For Novell networks, the default time for releasing locks made from a lost connection is five minutes. For other networks, your system administrator should be able to tell you how long the NOS takes to release the lock. In any case, the system administrator should be able to change this default release time to satisfy your particular needs.

If the licensing database has been installed on a stand-alone machine, the locks are immediately released and will again be available when the application is started again.

## ***License Distribution Options***

Because the ImageBASIC software is used in two distinct environments, development and runtime, different distribution methods are available to simplify the installation of licenses in each of these sites. As far as the license verification algorithms in the ImageBASIC software are concerned, the application can run as long as it is licensed. The following sections explain the different ways that ImageBASIC can be licensed.

### ***Development Licenses***

The standard licensing system for development is through a hardware key that plugs into any parallel port on the development machine. The hardware key separately enables each ImageBASIC component. *The hardware key is the default method for distribution of development licenses*

Under certain circumstances, developers may be unable to use a hardware key. The developer can install a licensing database that contains development licenses. Installing this database and changing the database or the hardware key require the developer to run the Licensing Configuration Manager and to call Diamond Head to receive an authorization code that enables the installation or change.

### ***Runtime Licenses***

For your clients who are running your compiled application, the same two sources of licensing authorization (the hardware key and the licensing database) are also available. Both have two primary control elements: a configuration program and an authorization program. The configuration program installs the licenses and returns a unique identification string for that installation. The authorization program provides a corresponding key string to complete the installation.

Diamond Head can provide you with the application that generates the authorization code. Because the developer will be able to generate authorization codes for the installation of a runtime licensing database, these clients will call the developer for the code, and Diamond Head Software need not be directly involved.

An option called Component Licensing is available for special circumstances that make the installation of a licensing database or the distribution of a hardware key unusually difficult. In Component Licensing, the application is authorized as it is compiled. It will then run without a hardware key or licensing database.



# Chapter 2 : Scanning

---

## Introduction to Scanning

The conversion of paper documents to make storage and retrieval of the information they hold more efficient and timely begins with the scanning process. Once the documents are captured in digital form, they may be further processed to enhance the appearance of the images, to extract the important data on the documents, and to store the images in a readily accessible format.

Beginning with the PixScan control to perform the initial capture, ImageBASIC offers controls to aid in the design and implementation of all steps in the storage and retrieval process.

## Programming Considerations

All scanning operations, particularly operations based on batch scanning, are processor intensive and require significant system resources. This means that you must carefully consider the available resources when designing your scanning routines. The main issues that must be considered when choosing hardware and when designing the application are

- Memory requirements
- Scanner capabilities
- CPU speed
- Disk drive access time and storage capacity
- Display speed

### ***Memory Required for Scanning, Display and Compression***

Images are considerably larger than most other types of data objects that are managed in batch processing.

- A data record may reach 1500 bytes
- An optimally compressed bitonal image file is generally 40,000 to 50,000 bytes *per page*.

The raw data required to represent a single letter size page scanned in black and white at 300 DPI is just under one megabyte. This data can be compressed in several different formats, each with its own advantages and disadvantages.

Because of its speed, Run Length Encoding (RLE) compression is applied to all image data held in memory by ImageBASIC. RLE compresses the image data to approximately twenty-five percent of its original size, or about 250 kilobytes. This file size must be considered when transferring images across a network.

The file sizes discussed above are for a bitonal, letter size image scanned at 300 DPI. Higher resolutions and grayscale or color images result in significantly larger volumes of data.

- For example, the same 8.5" X 11" page scanned at 300 DPI, but in 256 level grayscale instead of bitonal, requires almost eight megabytes of raw data to represent it.
- If RLE compression performed as well on grayscale data as it does on bitonal data, a single image will still require approximately two megabytes of RAM.
- Unfortunately, RLE does not usually provide this much compression with grayscale, so that same image will require up to four megabytes of RAM.

When an image is rotated, ImageBASIC creates a copy of the image in the new orientation, thereby doubling the required memory. Aside from the storage space required for the data, additional RAM is needed to perform the compression and decompression algorithms.

All of this required memory is in addition to that used by

- Microsoft Windows
- Network or communication drivers
- Display and print drivers
- The many other files held in memory by the operating system and other applications

After the total requirements for all parts of a scanning system are considered, it can be seen how an application that simply drives a scanner and writes to file will require a PC with at least eight megabytes of RAM. Any additional functionality--even just the addition of several more controls to the project, even if they are not used during scanning--will necessarily require more memory and a more powerful processor.

### ***Application Throughput vs. Scanner Speed***

It has been demonstrated that if you need to do more than you are doing now, it will take longer than it does now. If a program updates a database, adding the update of a second or third database or table will increase the time necessary for the program to complete. This same fact must be considered when designing a scanning routine. Consider these observations:

- As discussed above in "Memory Required for Scanning, Display and Compression," images are very large data objects. Recognizing this, applications must be designed to accommodate the larger object size.
- Scanning is a resource intensive task. A high speed scanner is capable of tying up a workstation all by itself, performing nothing but file saving, even without displaying the images as they are captured.
- The rated throughput of a scanner is determined by scanning alone, not in conjunction with database queries, network file transfers or image processing tasks.

The most common problem in scanner applications is being too complex. An application that scans, rotates images, does recognition of some kind, updates a database and writes files via network transfer is never going to approach the rated speed of high-end hardware. This is a critical point since scan throughput is normally the bottleneck in an imaging application. The key to a successful scanning application is to keep it simple.

- A scanning application should be limited as much as possible to just configuring and driving the scanner.
- Subsequent processing -- such as image enhancement, the extraction of indexing data through character or bar code recognition, form identification, etc. -- should be done in another program.
- In this way, scan throughput is maximized, and you gain the flexibility to process images independently of the scan operation, optimizing both processes.

---

## Scanner Hardware Setup

Scanners frequently ship with a driver that must be loaded in your CONFIG.SYS file, and the ImageBASIC scanner driver talks to this low level driver that came with your scanner.

- In the case of an SCSI scanner, the ASPI software usually provides the necessary interface layer.

Scanner run through other cards usually ship with different drivers:

- Some SCSI scanners ship with proprietary interface cards and require the system-level drivers supplied with them instead of ASPI.
- Scanners run through a Kofax board for example, will require the installation of KIPP software and sometimes a specific firmware driver for the scanner. In order for ImageBASIC to access the KIPP software, the KIPP/BIN directory must be on the PATH.

- Xionics driven scanners require the installation of PowerTools software. In order for ImageBASIC to access the PowerTools software, the PTOOLS/BIN directory must be on the PATH.

As you probably know by now, the process of installing hardware like scanners can stretch from 10 minutes to 6 hours, depending on the mood of the hardware gods on the day you are trying to do it. Try to be systematic and start from the beginning with the manufacturer-prescribed installation procedure.

Resolve any conflicts that you may have with the basic installation before moving on to the ImageBASIC dialog with the scanner. Once the scanner is properly installed, the rest of the process will work extremely well.

---

## Scanner Selection

Adding scanning capability to an ImageBASIC application is usually simple. ImageBASIC offers an interface with standard dialog boxes and easy routing of the scanned images to file or to a display control. The first step, however, is to make sure that your scanner is properly installed, using whatever setup software is provided by the manufacturer.

After the hardware is setup, ImageBASIC offers two methods of selecting the scanner that you have attached. Once the scanner is selected and ImageBASIC has verified communication with it, you are ready to begin scanning. Selecting a scanner may be accomplished through either a built-in dialog or through a series of methods and properties. Both of these options are detailed below.

### Scanner Selection -- Graphical

A standard scanner selection dialog is available for choosing the model of scanner that is being used. This dialog is opened by calling the **SelectScanner** method.

- The dialog box will display the names and models of all of the scanner drivers that are installed in the Windows/PIXTRAN directory.
- The user simply selects the correct model and clicks OK. The illustration below shows a typical selection dialog box.

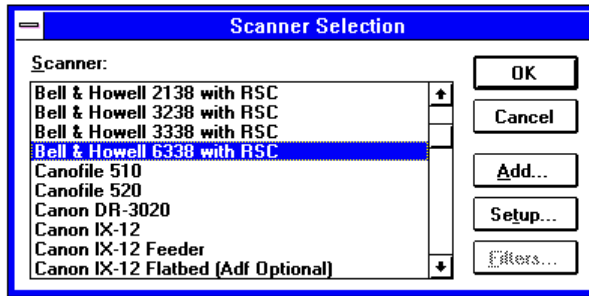


Figure 1 : Standard Scanner Selection dialog box listing all installed scanner drivers.

Some scanners require additional setup the first time that they are selected. If the scanner that you select requires additional information, the 'Setup...' button will be enabled. Click this button to review the available options such as the default page size and the enabling of installed hardware options

If the scanner is correctly attached and communication is established, ImageBASIC will load the scanner driver into memory and will set the **Scanner** property to the name of the scanner driver file.

**Note:** Kofax supported scanners are listed by model name and indicate that Kofax will be used. If using PowerTools, select the Xionics option in this dialog.

## Scanner Selection -- Programmatic

Each time a scanner driver is loaded into memory, an entry is made in SETSCAN.INI, and that scanner may be retrieved and used as the default.

If your application requires programmatic control of the selection and loading of scanner drivers, the following features of the PixScan control are available:

<b>Scanner</b> property	Specifies the name of the scanner driver
<b>GetSavedScanner</b> method	Returns the name of the most recently loaded scanner driver as recorded in SETSCAN.INI
<b>LoadScanner</b> method	Loads the scanner driver specified in the <b>Scanner</b> property
<b>UnloadScanner</b> method	Unloads the current scanner driver

### Specifying a Scanner Driver

To select a scanner without using the standard scanner selection dialog box, set the **Scanner** property to the name of the scanner driver. The entry for this property should be the name of the driver file (all scanner drivers have an extension of

.pxw), without the file extension. For example, to select a Kodak IL900D, set this property as shown here:

```
PixScan1.Scanner = "kodak"
```

Setting this property will not load the driver. The driver will be loaded when the **LoadScanner** method is executed. Selecting a scanner driver may also be performed through a built-in dialog box as discussed in "Scanner Selection -- Graphical" on page 8.

### ***Finding the Most Recently Used Scanner***

To determine the most recently used scanner, execute the **GetSavedScanner** method. This method will read SETSCAN.INI and return the name of the last scanner driver that was loaded. This method does not load the driver, nor does it change the value of the **Scanner** property.

```
scandrv$ = PixScan1.GetSavedScanner
```

If the PixScan control is instructed to load a scanner driver, configure the scanner, or scan a page without the user or application specifying a scanner, an error will occur. You must specify a scanner driver before performing any of these actions.

### ***Loading a Scanner Driver***

If the **Scanner** property is set to a valid value, the driver will be loaded by executing the **LoadScanner** method. The following code segment will find the most recently used scanner and attempt to load its driver:

```
PixScan1.Scanner = PixScan1.GetSavedScanner  
PixScan1.LoadScanner
```

A scanner driver must be loaded before any attempt to query or set scan options (through the standard dialog or through the tag interface) or start scanning.

### ***Unloading a Scanner Driver***

The **LoadScanner** method will load the scanner driver specified in the **Scanner** property. The scanner driver will be unloaded when the application is ended or when the **UnloadScanner** method is executed.

```
PixScan1.UnloadScanner
```

---

## Scanner Setup

All scanners offer some degree of configuration for the scanning process. This may be as simple as setting the page size or as complex as activating and optimizing a scanner-mounted image processing board.

The features of each scanner can be accessed either through a built-in dialog that is customized for each scanner or through a series of tagseach one of which addresses a single feature of the scanner.

- The built-in dialog offers the speed and convenience of a standard, easily accessible interface that presents all of the hardware options to the user in a compact form.
- The tag interface allows you to select which scanner options the user can modify, perhaps even perform all of the setup without any user interaction at all.

## Scanner Setup -- Graphical

A standard dialog box is supplied to allow the user to set the options available for the selected scanner. This dialog box is displayed by calling the **SetupScanner** method:

```
PixScan1.SetupScanner
```

The scanner setup dialog is slightly different for each scanner model, depending upon the features available in the hardware. Almost all scanners have options for resolution, page size, brightness and contrast, but many models offer more options. A typical scanner setup dialog box is shown below.

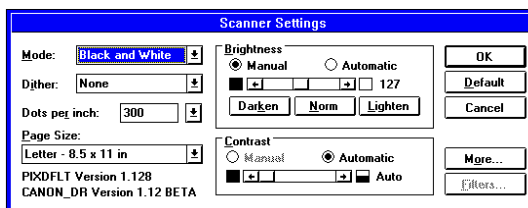


Figure 2 : Standard Scanner Setup dialog box showing basic options and driver versions.

Because of the large volume of information that must be presented on some scanner models, the scanner setup dialog box may have a 'More...' or an 'Area...' button enabled. Click these buttons to access advanced features of the selected scanner. Following is a typical example of a dialog opened by the 'More...' button:

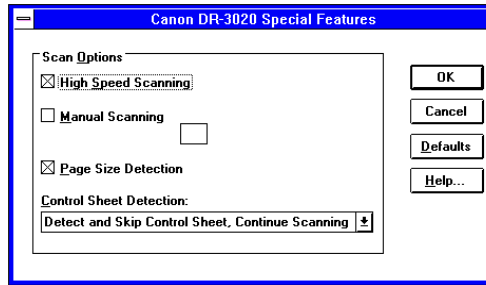


Figure 3 : Scanner Setup 'More...' dialog showing advanced options.

**Note:** The scanner setup dialog cannot be displayed properly unless the scanner driver is successfully loaded. The scanner driver can be loaded only when the scanner is correctly installed and the ImageBASIC license for scanning can be found.

## Scanner Setup -- Programmatic

The programmatic configuration of the scanner is accomplished through a tag interface. Each detail of scanner configuration -- such as page size, vertical and horizontal resolution, simplex or duplex scanning, and many others -- is represented by a tag. The PixScan control has several methods for reading and writing these tags.

The methods used to access the scanner control tags are as follows:

GetTagType	Reports data type of a tag -- long, string or rational
GetTagLength	Reports the number of indexed elements in a tag which is 1 (one) for most tags
SetTagDefault	Sets the specified tag to its default value

GetTagLong	Reports the current value of a tag of long data type
GetTagLongDefault	Reports the default for a tag of long data type
SetTagLong	Sets a new value for a tag of long data type

GetTagRational	Reports current value of a tag of rational data type
GetTagRationalDefault	Reports the default for the specified rational tag
SetTagRational	Sets a new value for a tag of rational data type

GetTagString	Reports the current value of the specified string tag.
--------------	--



SetTagString                      Sets a new value for a tag of string data type

If the application is designed with a custom scanner setup interface or makes configuration changes automatically, this tag interface must be used. Because of the number of available tags -- at this writing there are over 200 -- "Appendix B : Scanner Control Tags" on page 49 details their use.

## **Scanner Compression**

Many high performance scanners are capable of outputting compressed data to the host application. This feature helps to ensure the maximum throughput for the scanner. ImageBASIC defaults to enabling this onboard compression whenever the current scanner is so equipped. In some circumstances, it is necessary to disable the onboard compression performed by a scanner. For this purpose, the **UseScannerCompression** property is available:

- If **UseScannerCompression** is True, the default, ImageBASIC will query the scanner to determine if it is capable of outputting compressed data. If the scanner is capable, the compression will be enabled.
- If **UseScannerCompression** is False, ImageBASIC will instruct the scanner to output only uncompressed image data. With the exception of some very high performance scanners, this should not cause a noticeable change in the scanner throughput.

**Note:** Enabling or disabling the scanner's compression will not affect the compression of image files written from ImageBASIC. Only the transfer of image data from the scanner to the host application is affected.

## **Saving and Loading a Scanner Setup**

The current configuration of a scanner can be saved to file and then loaded at a later time to return the scanner to that configuration. This option may be used to save the last configuration so that when the operator returns, the same settings will be available, or this option may be used to store a set of configurations needed for different types of documents or forms. The **SaveTagSettings** method is used to save a scanner configuration.

Any of these saved configurations may be loaded at a later time and the scanner set to the saved configuration using the **RestoreTagSettings** method.

### **Save a Scanner Setup**

To save the current configuration, execute the **SaveTagSettings** method. This method will write an INI-like file containing the current tags settings for the active scanner. A scanner driver must be loaded to successfully execute this command.

The **SaveTagSettings** method accepts two string arguments, the name of the section to write and the file to write:

```
PixScan1.SaveTagSettings section_name , file_name
```

These arguments are optional.

- If no section name is specified, the name of the current scanner – as reported in the **Scanner** property – will be used as the section name.
- If no file name is supplied, SETSCAN.INI will be updated. This file is already used by ImageBASIC to store a limited amount of scanner information and is located in the Windows directory.

**Note:** If no section or file name is specified, only a single configuration for any given scanner can be saved. To save multiple configurations, use either separate sections within a single file or use separate files.

## ***Load a Scanner Setup***

After saving a scanner configuration using the **SaveTagSettings** method, the configuration may be loaded and the scanner set to the saved configuration by executing the **RestoreTagSettings** method. Note that the correct scanner driver must already be loaded using either the dialog opened by the **SelectScanner** method or using the **LoadScanner** method.

The **RestoreTagSettings** method accepts two string parameters, the name of the section to read and the name of the file to read:

```
PixScan1.RestoreTagSettings section_name , file_name
```

These arguments are optional.

- If no section name is specified, the name of the current scanner – as reported in the **Scanner** property – will be used as the section name.
- If no file name is supplied, SETSCAN.INI will be read.

---

## Selecting an Image Destination

When documents are scanned, the images can be sent to another ImageBASIC control for display or processing, or they can be sent directly to file. These two options are not mutually exclusive, however, and images can be simultaneously saved to file and passed to any ImageBASIC control as detailed in the following sections.

### Scanning to Another ImageBASIC Control

As image data is created by the scanner and passed to the PixScan control, that data can be directly routed to any other ImageBASIC control for display, enhancement, or other processing. However, when designing an application, keep in mind that scanning is a processor intensive task and any additional load placed on the CPU will negatively impact the performance of all tasks.

Refer to "Scanning One Page" on page 17 and "Scanning a Batch" on page 17 for details on beginning the scanning process.

#### Displaying Images During Scanning

To pass image data from a PixScan control to another ImageBASIC control as scanning is being performed, set the **ImageDataSource** of the recipient control to the PixScan control.

For example, you may wish to display each image in a TMS Display control as it is scanned. Create the link between the controls before scanning begins by executing this command:

```
TMSDispl.ImageDataSource = PixScan1
```

As new image files are generated by the scanner, the image data will be forwarded to the TMS Display control. As each new image arrives, the display control's **ImageDataChanged** event will be triggered. In this event, code can be executed to automatically perform any necessary processing on each image.

### Scanning to File

The PixScan control will save each image as it is generated. Refer to "Scanning One Page" on page 17 and "Scanning a Batch" on page 17 for details on beginning the scanning process.

The following properties control the file location and format:

OutputToFile	If True, the new image will be written to the file specified in the following properties; if False, no file will be created
--------------	---

OutputFileFormat	Specifies the output compression type and file type
OutputFileName	Specifies the fully qualified path and file name
OutputFileAppend	If True and the output file already exists, the next image will be appended to the end of the file. If False, the output file will be overwritten if it exists. Only TIFF, DCX and PDA files support multiple pages.

### 1) Setting the Output File Name

If the **OutputToFile** property is True, causing the PixScan control to write directly to file, the **OutputFileName** property must be set to the path and file name to write. The path and file name may be fully qualified or relative.

**Note:** The file name that is specified here can be the fully qualified file name, including drive letter. If the path is not supplied, the file will be written in the current directory.

```
PixScan1.OutputFileName = "c:\data\output\010101.tif"
```

### 2) Setting the File Format

The **OutputFileFormat** property specifies the format of the file written by the scanning control. The PixScan control supports the following file formats.

- 0 BMP Compressed
- 1 BMP Uncompressed
- 2 CALS Type1 Group 4
- 4 DCX (multi-page PCX)
- 5 GIF
- 7 JPEG
- 12 PCX
- 13 PDA G3
- 15 PDA G4
- 17 PDA Uncompressed
- 19 TIFF Group 3
- 20 TIFF Group 3 1-D
- 24 TIFF Group 4
- 26 TIFF LZW
- 27 TIFF Packed
- 28 TIFF Uncompressed

For example, to create TIFF files using CCITT Group 3 compression, set the **OutputFileFormat** property as shown here:

```
PixScan1.OutputFileFormat = 19
```

### 3) Creating Multiple Page Files

Several of the supported file formats allow the creation of a single file that contains multiple images. The **OutputFileAppend** property specifies whether the PixScan control will attempt to create these multi-page files.

- If the **OutputFileAppend** property is True and the **OutputFileName** property specifies a file that already exists, the next image will be added to the end of that file.
- If the specified file does not exist, it will be created.

**Note:** Only TIFF, PDA, and DCX file types support multiple pages in a single file. The **OutputFileAppend** property will be ignored for other file types, and any existing file will be overwritten.

## Scanning One Page

To scan a single page into the PixScan control, execute the **ScanPage** method.

- This function will not load the scanner driver.  
The scanner driver will be loaded when selected through the **SelectScanner** dialog or when the **LoadScanner** method is executed after setting the **Scanner** property.
- If the scanner supports page detection, the **IsPageLoaded** method may be executed to determine if a page is present. **Caution:** Some scanners will pull a page through the feeder to determine if one is present. This will not scan the page.
- After the image is received by the PixScan control, it will be written to a file if the **OutputToFile** property is True.
- The image can also be sent to another ImageBASIC control by setting the recipient control's **ImageDataSource** property to the PixScan control. Refer to "Scanning to Another ImageBASIC Control" on page 15 for details.

For example, you may wish to display each image as it is scanned. To do this, link an ImageBASIC display control to the scan control:

```
TMSDispl1.ImageDataSource = PixScan1.Link
```

## Scanning a Batch

Batch scanning is the normal procedure in a production scanning operation. Instead of scanning one page at a time, a stack or group of documents is fed into the scanner, and all the user has to do is start the scan. For this type of application, ImageBASIC offers the **ScanBatch** method.

If the scanner supports page detection, the **IsPageLoaded** method may be executed to determine if a page is present. **Caution:** Some scanners will pull a page through the feeder to determine if one is present. This will not scan the page.

**ScanBatch** automatically activates scan ahead buffering in scanners that support it, and it has the added speed advantage of initializing the scanner only once rather than on each page. Any changes in the destination of the images -- for example, creating a new file name for each page -- must be programmed into the scanning events, **WillScan** and **DidScan**.

**ScanBatch** is a very high performance function that will continue scanning until one of four things happen:

- The scanner runs out of input
- A scan returns no image data
- A scanner error is returned
- The **CancelScan** flag is set to True in the **WillScan** event

To begin a batch scan, set the output options as desired and then call the **ScanBatch** method:

```
PixScan1.OutputToFile = True
PixScan1.OutputFileType = 24 ' TIFF Group 4
PixScan1.OutputFileAppend = True
PixScan1.OutputFileName = "c:\scanned\batch01\2001.tif"
PixScan1.ScanBatch
```

## ***Stopping a Batch Scan***

Once the **ScanBatch** method is called, the only way the application can stop the process is by setting the **CancelScan** flag to True in the **WillScan** event.

- A common technique of canceling a batch scan is to display or enable a button during the scan.
- Clicking on this button sets a global variable to True, and the **CancelScan** parameter is set equal to this global variable in the **WillScan** event.

As an example of canceling a batch scanning operation, assume the following:

a global variable named *gnStopScan*

a button named *cmdStopScan*

When the button is clicked, execute code similar to the following:

```

Private Sub cmdStopScan_Click ()
    gnStopScan = True
End Sub

```

In the **WillScan** event, set the *CancelScan* parameter to stop the batch:

```

Sub PixScan1_WillScan (PageIndex as Integer, CancelScan as
    Boolean)

    DoEvents      ' To process the button click

    CancelScan = gnStopScan

    PixScan1.ClearScanAhead

End Sub

```

Any pages that have been scanned but not processed will be held in the scan ahead buffers. If another scan operation is begun, these pages in the scan ahead buffers will be read before the scanner begins again. Calling the **ClearScanAhead** method as shown in the code above will remove these images from the buffer, allowing the first page on the scanner to be the first one read when the next scan operation begins.

## Scan Ahead Pages

In order to optimize performance in a batch scanning process, ImageBASIC will accept images from the scanner and store them in memory before the Pixel Scan control has resources to process them. These pre-scanned pages are commonly referred to as scan ahead pages. Any image stored in the scan ahead buffers will be processed before new images are received from the scanner.

These extra pages are generally only an issue when a batch scan is interrupted by the *CancelScan* flag during the **WillScan** event. When the next scan function is called, the images in the buffers will be read first. If you prefer to scan the next page on the scanner without processing the images in the scan buffers, the extra images must be removed from memory. The **ClearScanAhead** method may be called at any time to flush all images from the scan ahead buffers.

---

## Scanning Events

Whenever a PixScan control is scanning a document, two events occur for each page:

WillScan

DidScan

### The WillScan Event

The **WillScan** event occurs immediately before each page is scanned. For the first page in a batch scan, this event occurs after driver initialization. This event provides the following parameters:

PageIndex	Reports the number of the page that is about to be scanned. Begins at 1 (one).
CancelScan	If set to True during this event, the batch will be canceled. Refer to "Stopping a Batch Scan" on page 18 for details.

Any changes to the scan destination or the output file name must be made in the **WillScan** event or before scanning is begun.

### The DidScan Event

The **DidScan** event occurs after image data has been output from the PixScan control; i.e., after the file has been saved if **OutputToFile** is True and after any ImageBASIC control that is linked to the PixScan control has received the image. This event provides one parameter:

PageIndex	Reports the number of the page that was just scanned.
-----------	---



# Chapter 3 : Reference

---

## Reference of Properties, Events and Methods

The following pages contain a technical reference to the properties, methods and event of this control.

### ***AboutBox Method***

<b>Definition:</b>	Displays a message box showing version and copyright information when queried.
<b>Syntax:</b>	<code>PixScan1 .AboutBox</code>
<b>Return Values:</b>	None
<b>Description:</b>	The message box is application modal and has a single OK button on it. The message box is unloaded when the button is clicked.

### ***Active Property***

<b>Definition:</b>	<p>If set to True at design time, the control will fully initialize and verify licensing immediately upon initialization of the runtime application.</p> <p>If set to False at design time, full initialization of the control will be delayed at initialization of the runtime application. In this case, this property must be explicitly set to True at runtime before the control is used.</p>
<b>Data Type:</b>	Boolean
<b>Design Access:</b>	Read/Write
<b>Runtime Access:</b>	Read/Write (see limits below)
<b>See Also:</b>	"Licensing Configuration and Verification" on page 2
<b>Comments:</b>	<p>If this property is set to True (the default) at design time, the control is fully initialized and licensing is verified immediately upon initialization of the application at runtime. The technology libraries are loaded and the control is ready to be used.</p> <p>If this property is set to False at design time, the control will only partially initialize when the application loads at runtime. By performing the two actions shown below, the application should be able to load more quickly:</p> <ol style="list-style-type: none"><li>1) The technology libraries for the control will not be loaded.</li><li>2) The licensing server will not find a token for the control.</li></ol>

If the control initializes with **Active** set to False, this property must be explicitly set to True by the application. Until **Active** is set to True, the control will ignore all instructions to it.

When the **Active** property is set to True, the two delayed actions – loading the technology libraries and verifying a license – will be performed. Some lag will be noticed at this point, but this delay occurs only once.

If the control fails to find a license token, the **Active** property will be automatically set to False. The application can check this value on Form Load or later to find if the control is licensed.

### ***Clear Method***

<b>Definition:</b>	Clears the currently loaded image from the control.
<b>Parameters:</b>	None
<b>Syntax:</b>	<code>PixScan1.Clear</code>
<b>Return Value:</b>	0 on failure 1 on success
<b>Data Type:</b>	Long
<b>See Also:</b>	ClearScanAhead Method
<b>Comments:</b>	When the currently loaded image (which is usually the last scanned image) is cleared, all controls receiving image data from this control will also be cleared.  For example, if performing a batch scan to a display control, the last image in the batch will be maintained in the PixScan control after scanning is complete. The <b>Clear</b> method will remove this image from memory and consequently clear the Display control.

### ***ClearScanAhead Method***

<b>Definition:</b>	Clears the scan ahead buffers of all images.
<b>Parameters:</b>	None
<b>Syntax:</b>	<code>nRet = PixScan1.ClearScanAhead</code>
<b>Return Value:</b>	0 on failure or if no pages in buffer Number of pages discarded on success
<b>Data Type:</b>	Long
<b>See Also:</b>	WillScan Event, ScanBatch Method
<b>Comments:</b>	When a batch scanning operation is halted by anything other than the scanner running out of paper, some images may remain in the scan ahead buffers. If these images are left in the buffers, they will be read and processed during the next scan operation.

## ***ConfigureScanner Method***

<b>Definition:</b>	Opens the scanner configuration dialog.
<b>Parameters:</b>	None
<b>Syntax:</b>	<code>PixScan1.ConfigureScanner</code>
<b>Return Value:</b>	0 on failure or 'Cancel' button clicked 1 on success
<b>Data Type:</b>	Long
<b>See Also:</b>	SetupScanner Method
<b>Comments:</b>	The scanner configuration dialog is usually accessed by clicking the 'Configure...' button in the scanner setup dialog. The options in this dialog are generally advanced or scanner specific. This dialog may not be available for all scanner models.

## ***DidScan Event***

<b>Definition:</b>	Occurs after each page is scanned and image output is complete.
<b>Parameters:</b>	PageIndex
<b>See Also:</b>	WillScan Event, ScanBatch Method
<b>Comments:</b>	This event occurs after the output file is created if <b>OutputToFile</b> is True and after any other ImageBASIC control that is linked to the scanner control receives the image data.

## ***Error Event***

<b>Definition:</b>	Occurs for any error internal to this control.	
<b>Parameters:</b>	See Below	
<b>Comments:</b>	This event occurs only for errors internal to the specific control. If no code is placed in this event, the standard dialog is shown and displays the same message that is reported in the Description parameter.	
	Number	Reports the error number
	Description	Reports a descriptive string of the error
	SCode	ODBC error return code
	Source	Reports a string of the source of the error
	HelpFile	Reports the name of a help file that should explain this error
	HelpContext	Reports the help context ID in the HelpFile that matches this error
	CancelDisplay	If set to 0 in this event, the standard error reporting dialog will not be displayed

### ***GetSavedScanner Method***

<b>Definition:</b>	Returns the name of the last scanner driver loaded as it is recorded in setscan.ini.
<b>Parameters:</b>	None
<b>Syntax:</b>	<code>sScanName = PixScan1.GetSavedScanner</code>
<b>Return Value:</b>	String of scanner driver name on success
<b>Data Type:</b>	Long
<b>See Also:</b>	Scanner Property, LoadScanner Method, SelectScanner Method
<b>Comments:</b>	Successful execution of this method does not change the currently loaded scanner driver, if any, and does not set the <b>Scanner</b> property.

### ***GetTagChoiceCount Method***

<b>Definition:</b>	Returns a long number totaling the number of legal values for the specified tag.
<b>Parameters:</b>	Tag            ID of tag to query
<b>Syntax:</b>	<code>nVal = PixScan1.GetTagChoiceCount( tagID )</code>
<b>Return Value:</b>	Long number of count of legal values
<b>See Also:</b>	GetTagChoiceFlags Method, GetTagChoiceString Method, GetTagChoiceRational Method, GetTagChoiceLong Method
<b>Comments:</b>	This method will return the number of legal values for a tag even if the tag is a range type. For example, TAG_BRIGHTNESS has a range of legal values from 0 to 255 on some scanners. This method will have a return value of 256.

### ***GetTagChoiceFlags Method***

<b>Definition:</b>	Returns a long value of the type of tag -- range or list.
<b>Parameters:</b>	Tag            ID of tag to query
<b>Syntax:</b>	<code>nVal = PixScan1.GetTagChoiceFlags( tagID )</code>
<b>Return Value:</b>	2            Range Type 4            List Type
<b>See Also:</b>	GetTagChoiceCount Method
<b>Comments:</b>	A range type tag has values represented internally as a high value, a low value, and a step value. A list type tag has values represented as a fix list of the valid options.  The return value from this method should be compared to the possible return values with the AND operand. For example, to determine the tag type, use the following pseudo-code:

```
TagFlags = PixScan1.GetTagChoiceFlags( tagID )
```

```

If (TagFlags AND 2) Then
    * * * tag is range type * * *
Else
    If (TagFlags AND 4) Then
        * * * tag is list type * * *
    Else
        * * * tag is neither range nor list type
    End If
End If

```

### ***GetTagChoiceLong Method***

**Definition:** Reports the value of one legal option in a tag's list of legal values for all tags accepting long data type values.

**Parameters**

Tag	ID of tag to query
Index	Position of value to query in the tag's list

**Syntax:** `nRet = PixScan1.GetTagChoiceLong( tagID, index)`

**Return Value:** Long value of the option at the specified index

**See Also:** GetTagChoiceCount Method, GetTagType Method, SetTgLong Method

**Comments:** This method may be used only with tags that are of the long data type.

All numerical tags, whether they are enumerated (list type) or have a range of values (range type), maintain a list of all of their legal values. Each legal value is assigned an index in that list. This method reads the value assigned to an index, allowing the application to find all of the legal values for any tag that accepts long values.

For example, the following Visual Basic code reads all of the values for TAG\_PHOTOMETRICINTERPRETATION (106 Hex) that are supported on the currently selected scanner and displays them in a list box.

```

Dim nCount as Integer
Dim lTagOpt as Long
nCount = PixScan1.GetTagChoiceCount(&H106)
For i = 0 to valCount - 1
    lTagOpt = PixScan.GetTagChoiceLong(&H106, i)
    List1.AddItem lTagOpt
Next i

```

Refer to "Appendix B : Scanner Control Tags" on page 49 for details on supported tags and their data types.

### ***GetTagChoiceRational Method***

- Definition:** Reports the value of one legal option in a tag's list of legal values for all tags accepting rational values.
- Parameters**
- |       |  |
|-------|--|
| Tag   | ID of tag to query                           |
| Index | Position of value to query in the tag's list |
- Syntax:** `wRet = PixScan1.GetTagChoiceRational( ID, index )`
- Return Value:** Double (4 byte) numerical value of the option at the specified index
- See Also:** GetTagChoiceCount Method, GetTagType Method, SetTagRational Method
- Comments:** This method may be used only with rational value type tags. All numerical tags, whether they are enumerated (list type) or have a range of values (range type), maintain a list of all of their legal values. Each legal value is assigned an index in that list. This method reads the value assigned to an index, allowing the application to find all of the legal values for any tag that accepts rational values. TAG\_XRESOLUTION and TAG\_YRESOLUTION are the most commonly used rational tags.

### ***GetTagChoiceString Method***

- Definition:** Reads the legal value for a tag that has been assigned the specified index value. Valid only for string tags.
- Parameters**
- |        |  |
|--------|--|
| Tag    | ID of the tag to query                 |
| Index  | Index position of the value to query   |
| Length | Maximum number of characters to return |
- Syntax:** `sRet = PixScan1.GetTagChoiceString( ID, 0, 256 )`
- Return Value:** String value of the index on success
- See Also:** GetTagType Method, SetTagString Method, "Appendix B : Scanner Control Tags" on page 49
- Comments:** Each scanner driver maintains a list of all valid options for every supported tag. This method allows the user to find the legal values for the tag.
- For example, the following Visual Basic code segment will populate a list box with all of the options for TAG\_PAGESIZE (50E Hex), accepting up to 40 characters for each option.
- ```
Dim nVal as Integer
Dim i as Integer
```

```

nVal = PixScan1.GetTagChoiceCount(&H50E)
For i = 0 to (nVal - 1)
    Opt = PixScan1.GetTagChoiceString(&H50E, i, 40)
    List1.AddItem Opt
Next i

```

### ***GetTagLength Method***

**Definition:** Reports the number of indexed elements in a tag.

**Parameters** Tag ID of tag to query

**Syntax:** `nVal = PixScan1.GetTagLength( tagID)`

**Return Value:** Long number indicating the number of elements in the tag

**See Also:** GetTagType Method, SetTagLong Method, SetTagString method, SetTagRational Method

**Comments:** A few tags maintain an array of values. This method queries the tag to find out how many elements are in the array. However, most tags hold only one value, and for these tags, this method will return a value of 1 (one).  
If an ASCII tag is queried, the number of characters in the current setting will be returned.

### ***GetTagLengthDefault Method***

**Definition:** Reports the number of indexed elements for the tag's default value.

**Parameters** Tag ID of tag to query

**Syntax:** `nVal = PixScan1.GetTagLengthDefault( tagID)`

**Return Value:** Long number indicating the number of elements in the tag

**Comments:** A few tags maintain an array of values. This method queries the tag to find out how many elements are in the array. However, most tags hold only one value, and for these tags, this method will return a value of 1 (one).  
If an ASCII tag is queried, the number of characters in the default setting will be returned.

### ***GetTagLong Method***

**Definition:** Reads the current value of the specified long tag.

**Parameters** Tag ID of tag to query  
Index Index into tag to query, usually 0 (zero)

**Syntax:** `lVal = PixScan1.GetTagLong( tagID, index)`

**Return Value:** Long number indicating the current value of the specified tag element

**See Also:** GetTagString Method, GetTagRational Method, GetTagType Method

**Comments:** Most tags hold only one value at a time and will return a meaningful value for only Index = 0. This method can only read the values for tags that are short (two byte) or long (four byte) data types.

### ***GetTagLongDefault Method***

**Definition:** Reports the default value for the specified element of a tag.

**Parameters**

|       |                                         |
|-------|-----------------------------------------|
| Tag   | ID of tag to query                      |
| Index | Index in tag to query, usually 0 (zero) |

**Syntax:** `lVal = PixScan1.GetTagLongDefault( tagID, index)`

**Return Value:** Long number indicating the default value of the specified index

**See Also:** GetTagLong Method, GetTagType Method, SetTagDefault Method

**Comments:** This method will read only tags that maintain short or long data type values.

Most tags hold only one value at a time, because they have only one index element. Therefore, for most tags, this method will be called with the Index parameter set to 0 (zero).

### ***GetTagRational Method***

**Definition:** Reads the current value of the specified rational tag.

**Parameters**

|       |                                         |
|-------|-----------------------------------------|
| Tag   | ID of tag to query                      |
| Index | Index in tag to query, usually 0 (zero) |

**Syntax:** `lVal = PixScan1.GetTagRational( tagID, index)`

**Return Value:** Four-byte number indicating current value of the specified index in a tag

**See Also:** GetTagString Method, GetTagLong Method, GetTagType Method

**Comments:** This method will return the current value for rational tags only. The only two commonly used rational tags are TAG\_XRESOLUTION and TAG\_YRESOLUTION.

### ***GetTagRationalDefault Method***

**Definition:** Reports the default value for the specified rational tag.

**Parameters**

|       |                                         |
|-------|-----------------------------------------|
| Tag   | ID of tag to query                      |
| Index | Index in tag to query, usually 0 (zero) |

**Syntax:** `lVal = PixScan1.GetTagRationalDefault( tagID, 0)`



**Return Value:** Four-byte number indicating the default for the specified element in the tag

**See Also:** GetTagRational Method, SetTagDefault Method

**Comments:** All of the currently supported rational tags have only one indexed element, so this method will always be called with an Index of 0 (zero). The only rational tags that are commonly used are TAG\_XRESOLUTION and TAG\_YRESOLUTION.

### ***GetTagString Method***

**Definition:** Reads the current value of the specified string tag.

**Parameters** Tag ID of the tag to query  
Index Index element in the tag to query, usually 0 (zero)

**Syntax:** `sVal = PixScan1.GetTagString( tagID, index )`

**Return Value:** String of the current tag value

**See Also:** GetTagType Method, SetTagString Method

**Comments:** All of the currently supported string tags have only one indexed element, so this method will always be called with an Index of 0 (zero). The most commonly used string tag is TAG\_PAGESIZE. For example, the following will read the current value of index 0 (the only valid index as reported by the **GetTagLength** method) for the page size tag and display it in a label:  
`Label1.Caption = PixScan1.GetTagString(&H102, 0)`

### ***GetTagStringDefault Method***

**Definition:** Returns the default value of the specified string tag.

**Parameters** Tag ID of tag to query  
Length Maximum number of characters to return

**Syntax:** `sVal = PixScan1.GetTagStringDefault( tag, length )`

**Return Value:** 0 on failure  
String value of default on success

**See Also:** GetTagType Method, SetTagDefault Method, GetTagString Method

**Comments:** The default value for any string tag may be found using this method.

### ***GetTagType Method***

**Definition:** Reports the data type of the specified tag.

**Parameters** Tag ID of the tag to query

**Syntax:** `nVal = PixScan1.GetTagType( tagID )`

**Return Value:** 0 on failure

Enumerated value of tag type on success (see below)

**See Also:** GetTagLong Method, SetTagLong Method, GetTagString Method, SetTagString Method, GetTagRational Method, SetTagRational Method

**Comments:** The return value of this method will correspond to one of the following values:

- 1 Byte \*
- 2 ASCII
- 3 Short \*
- 4 Long \*
- 5 Rational

For the purposes of choosing the proper method to read and write tag values, all Byte, Short, and Long tags are considered Long.

### ***IsPageLoaded Method***

**Definition:** Queries the current scanner to determine if a page is loaded.

**Parameters:** None

**Syntax:** `nVal = PixScan1.IsPageLoaded`

**Return Value:**

- 1 page detection is not supported
- 0 page detection is supported, but no pages are present
- 1 page detection is supported, and a page is in the ADF
- 2 page detection is supported, and a page is on the flatbed

**See Also:** GetTagLong Method, TAG\_FEEDER

**Comments:** This method will only report a valid value for a scanner that supports an ADF. The scanner driver must be loaded before this method is executed. To load a scanner driver, execute the **SelectScanner** or **LoadScanner** method.

**Caution:** Some scanners must actually feed a page to determine if one is loaded. Feeding the page for this purpose will generally not scan the page, so execution of this method on this type of scanner will cause the loss of the first page in the ADF.

### ***Link Property***

**Definition:** Reports the Link ID calculated for this control at its creation.

**Data Type:** String

**Syntax:** `TMSDispl.ImageDataSource = PixScan1.Link`

**Design Access:** Not Available

**Runtime Access:** Read-only

**Comments:** Each ImageBASIC control is assigned a unique Link ID at its creation. This Link ID can be specified in the **ImageDataSource**, **DisplaySource**, **RegionSource**, and **AnnoteSource** properties of various ImageBASIC controls. These source properties specify the ImageBASIC control that is supplying information or services to a control.

### ***LoadScanner Method***

**Definition:** Loads the scanner driver specified in the **Scanner** property.

**Parameters:** None

**Syntax:** `PixScan1.LoadScanner`

**Return Value:** 1 on success  
0 on failure

**See Also:** Scanner Property, UnloadScanner Method

**Comments:** If another scanner driver is already loaded, it will be unloaded by this method. The loaded driver may also be unloaded by executing the **UnloadScanner** method.

The **Scanner** property may be explicitly set to any valid scanner driver name. The most recently used scanner driver can be found using the **GetSavedScanner** method.

**Note:** The specified scanner must be available and communicating properly to load the driver.

### ***OutputFileAppend Property***

**Definition:** If True, any image saved to file will be appended to the output file if it already exists.

**Data Type:** Boolean

**Design Access:** Read/Write

**Runtime Access:** Read/Write

**See Also:** OutputToFile Property, OutputFileName Property

**Possible Values:** True  
False

**Comments:** Some of the file formats supported by ImageBASIC do not support the creation of multi-page files. The file formats that can maintain multiple pages are TIFF, PDA, and DCX. Attempting to append to any other file will cause the file to be overwritten.

### ***OutputFileFormat Property***

**Definition:** Specifies the file format of the output file.

**Data Type:** Enumerated

**Design Access:** Read/Write  
**Runtime Access:** Read/Write  
**See Also:** OutputToFile Property, OutputFileName Property  
**Possible Values:** See below  
**Comments:** The following list shows the supported file formats and the values for this property:

- 0 BMP Compressed
- 1 BMP Uncompressed
- 2 CALS Type1 G4
- 4 DCX (multi-page PCX)
- 5 GIF
- 7 JPEG
- 12 PCX
- 13 PDA G3
- 15 PDA G4
- 17 PDA Uncompressed
- 19 TIFF G3
- 20 TIFF G3 1-D
- 24 TIFF G4
- 26 TIFF LZW
- 27 TIFF Packed
- 28 TIFF Uncompressed

**Note:** Different file format are limited in the level and type of color images that they support. Files saved by the PixScan control will be of the same color definition as the image received from the scanner. Therefore, make certain that the file format selected does support the specified color format. Refer to 'Color Limitations of File Formats' on page 46 for a detailed chart.

### ***OutputFileName Property***

**Definition:** Specifies the file name of the image file to save if the **OutputToFile** property is True.  
**Data Type:** String  
**Design Access:** Read/Write  
**Runtime Access:** Read/Write  
**See Also:** OutputToFile Property, OutputFileFormat Property, OutputFileAppend Property  
**Possible Values:** Any valid path and file name  
**Comments:** This property may be set to any valid path and file name. The path may be relative or absolute. As with most file name, the extension may be arbitrarily assigned, but it is recommended that

the standard file extensions be applied; e.g., *tif* for all TIFF files, *.jpg* for JPEG files, etc.

### ***OutputToFile Property***

- Definition:** If True, all scanned images will be saved to file. If False, scanned images will be loaded into memory and made available to any other ImageBASIC control whose **ImageDataSource** is set to the Name of this PixScan control.
- Data Type:** Boolean
- Design Access:** Read/Write
- Runtime Access:** Read/Write
- See Also:** OutputFileName Property, OutputFileFormat Property, OutputFileAppend Property
- Possible Values:** True  
False
- Comments:** All images will be passed to any other ImageBASIC control that is linked to the PixScan control (by setting the **ImageDataSource** of the recipient control). The image may also be saved to file by setting the **OutputToFile** property to True before scanning is begun.

### ***RestoreTagSettings Method***

- Definition:** Loads a scanner configuration saved by the **SaveTagSettings** method.
- Parameters**
- |                 |                                               |
|-----------------|-----------------------------------------------|
| <i>section</i>  | String name of the section to which to read   |
| <i>filename</i> | String path and file name of the file to read |
- Syntax:** `PixScan1.RestoreTagSettings section, filename`
- Return Value:** 0 on success  
Negative error code on failure
- See Also:** SaveTagSettings Method, "Scanner Setup" on page 11
- Comments:** The parameters to this method are optional. If the section name is omitted, the name of the current scanner, as reported in the **Scanner** property, will be used as the section name. If the file name is omitted, SETSCAN.INI in the Windows directory will be read.
- Note: The proper scanner driver must be loaded before this method is executed. Refer to the **LoadScanner** method and the **SelectScanner** method for details on selecting and loading a scanner driver.

## ***SaveTagSettings Method***

- Definition:** Saves the current scanner settings to an INI-like file of the specified name.
- Parameters:** *section* String name of the section to which to write the current scanner settings  
*filename* String path and file name of the file to write
- Syntax:** `PixScan1.SaveTagSettings section, filename`
- Return Value:** 0 on success  
Negative error code on failure
- See Also:** RestoreTagSettings Method, 'Scanner Setup' on page 11
- Comments:** The parameters to this method are optional. If the section name is omitted, the name of the current scanner, as reported in the **Scanner** property, will be used as the section name. If the file name is omitted, SETSCAN.INI in the Windows directory will be written.
- Only a subset of the supported scanner configuration tags will be saved. The exact set depends upon the scanner and driver that are being used. For this reason, it is strongly recommended that you test this feature using the setup that will be used in production.

## ***ScanAhead Property***

- Definition:** If True, the scanner will be instructed to use scan ahead buffers.
- Data Type:** Boolean
- Design Access:** Read/Write
- Runtime Access:** Read/Write
- Possible Values:** True (Default)  
False
- See Also:** ScanBatch Method
- Comments:** ImageBASIC utilizes scan ahead buffers to maximize performance whenever the scanner hardware allows. By accepting images from the scanner before they can actually be processed, the scan control does not make the scanner wait to be instructed to scan each page. Instead, the scanner will run at full speed and store image in the scan ahead buffer from which the PixScan control will retrieve them when it is ready.
- Most scanners with automatic paper feeders support the scan ahead feature. If the selected scanner does not, setting this property to True will not cause an error but does not improve performance.

## ***ScanBatch Method***

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Definition:</b>   | Initiates a new scan using the currently selected scanner driver and setup.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Parameters:</b>   | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Syntax:</b>       | <code>PixScan1.ScanBatch</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Return Value:</b> | 0 on failure<br>The number of pages scanned on success                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>See Also:</b>     | ScanPage Method, OutputToFile Property, SelectScanner Method, SetupScanner Method                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Comments:</b>     | <p>All images that are scanned are made available to other ImageBASIC controls. These controls must name the PixScan control in their <b>ImageDataSource</b> property to access the images.</p> <p>If the <b>OutputToFile</b> property is True, the scanned images will be saved directly to the file and in the format specified in these properties:</p> <ul style="list-style-type: none"><li>OutputFileName</li><li>OutputFileFormat</li><li>OutputFileAppend</li></ul> <p>The <b>WillScan</b> event occurs before each page is scanned; the <b>DidScan</b> event occurs after each page is scanned and saved. In these events, the <b>OutputFileName</b> or other output options may be set.</p> <p>A scanner driver must be specified before this method is executed. A scanner may be selected using the <b>SelectScanner</b> method or by setting the <b>Scanner</b> property to any valid driver name.</p> <p><b>Note:</b> Only scanners that can detect the presence of paper in their feeder support this method. Unless canceled, scanning continues until the scanner runs out of paper or an error occurs.</p> |

## ***Scanner Property***

|                         |                                                                                                                                                               |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Definition:</b>      | Specifies the name of the selected scanner driver.                                                                                                            |
| <b>Data Type:</b>       | String                                                                                                                                                        |
| <b>Design Access:</b>   | Read/Write                                                                                                                                                    |
| <b>Runtime Access:</b>  | Read/Write                                                                                                                                                    |
| <b>See Also:</b>        | LoadScanner Method, SelectScanner Method                                                                                                                      |
| <b>Possible Values:</b> | Any valid scanner driver name                                                                                                                                 |
| <b>Comments:</b>        | This property may be explicitly set to the name of a valid scanner driver, and the driver will then be loaded when the <b>LoadScanner</b> method is executed. |

In order to load the most recently used scanner driver, the **GetSavedScanner** method may be used. This method reads SETSCAN.INI, in which scanner selection is recorded, and returns the name of the most recent driver.

### ***ScanPage Method***

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Definition:</b>   | Instructs the currently selected scanner to scan one page.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Parameters</b>    | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Syntax:</b>       | <code>PixScan1.ScanPage</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Return Value:</b> | 0 on failure<br>1 on success                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>See Also:</b>     | ScanBatch Method, SelectScanner Method, SetupScanner Method, OutputToFile Property                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Comments:</b>     | <p>A scanner driver must be selected before scanning is initiated. Refer to the <b>Scanner</b> property or the <b>SelectScanner</b> method for details on selecting a scanner driver.</p> <p>All images that are scanned are made available to other ImageBASIC controls. These controls must name the PixScan control in their <b>ImageDataSource</b> property to access the images.</p> <p>If the <b>OutputToFile</b> property is True, the scanned images will be saved directly to the file and in the format specified in these properties:</p> <ul style="list-style-type: none"><li>OutputFileName</li><li>OutputFileFormat</li><li>OutputFileAppend</li></ul> <p>The <b>WillScan</b> event occurs before each page is scanned; the <b>DidScan</b> event occurs after each page is scanned and saved. In these events, the <b>OutputFileName</b> or other output options may be set.</p> |

### ***SelectScanner Method***

|                      |                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Definition:</b>   | Opens a standard scanner selection dialog box listing all of the installed scanner drivers.                                                                                                  |
| <b>Parameters</b>    | None                                                                                                                                                                                         |
| <b>Syntax:</b>       | <code>PixScan1.SelectScanner</code>                                                                                                                                                          |
| <b>Return Value:</b> | 0 on failure or 'Cancel'<br>1 on success                                                                                                                                                     |
| <b>See Also:</b>     | SetupScanner Method, Scanner Property                                                                                                                                                        |
| <b>Comments:</b>     | Selecting a scanner in this dialog sets the <b>Scanner</b> property but does not load the driver. The driver will be loaded when required or when the <b>LoadScanner</b> method is executed. |



**Note:** Most scanners are identified in this dialog by make and model. However, when using any scanner with a Xionics interface card, choose the Xionics option in this dialog.

### ***SetTagDefault Method***

**Definition:** Sets the specified tag to its default value.

**Parameters** Tag ID of the tag to set

**Syntax:** `PixScan1.SetTagDefault tagID`

**Return Value:** 0 on failure  
1 on success

**See Also:** GetTagType Method

**Comments:** Will set any data type of tag -- long, rational, string -- to its default value. If this method is called specifying a tag ID of 0 (zero), all tags will be set to defaults.

### ***SetTagLong Method***

**Definition:** Sets the indexed tag element to the specified long tag.

**Parameters** Tag ID of the tag to set  
Index Index into the tag to set  
Value Value to which to settag

**Syntax:** `PixScan1.SetTagLong tagID, index, value`

**Return Value:** 0 on failure  
1 on success

**See Also:** GetTagLong Method, SetTagRational Method, SetTagString Method, GetTagType Method

**Comments:** Scanner control tags are of different data types -- byte, short, long, ASCII (string), or rational. The **SetTagLong** method is used to set byte, short, and long tags. Refer to 'Appendix B : Scanner Control Tags' on page 49 for details on the scanner control tags that are addressed through this method.

### ***SetTagRational Method***

**Definition:** Sets the indexed tag element to the specified value. Must be a rational tag.

**Parameters** Tag ID of the tag to set  
Index Index into the tag to set, usually 0 (zero)  
Value Value to which to set tag

**Syntax:** `PixScan1.SetTagRational tagID, index, value`

**Return Value:** 0 on failure  
1 on success

**See Also:** SetTagLong Method, SetTagString Method, GetTagType Method

**Comments:** Scanner control tags are of different data types -- byte, short, long, ASCII (string), or rational. The **SetTagRational** method is used to set only rational tags. Refer to "Appendix B : Scanner Control Tags" on page 49 for details on the scanner control tags that are addressed through this method.

Either the **SetTagRational** or the **SetTagRational2** method may be used to set any rational tag. The difference between the two methods is that the true value must be calculated for **SetTagRational** while **SetTagRational2** accepts the numerator and denominator as separate values. For example, both of the following lines of code set the X-resolution to 300 DPI:

```
PixScan1.SetTagRational &H11A, 0, 65836
PixScan1.SetTagRational2 &H11A, 0, 300, 1
```

### ***SetTagRational2 Method***

**Definition:** Sets the indexed tag element to the specified value. Must be a rational tag.

**Parameters:**

|        |                                               |
|--------|-----------------------------------------------|
| Tag    | ID of the tag to set                          |
| Index  | Index into the tag to set, usually 0 (zero)   |
| ValNum | Value of numerator number in rational value   |
| ValDen | Value of denominator number in rational value |

**Syntax:** `PixScan1.SetTagRational2 tagID, index, num, den`

**Return Value:** 0 on failure  
1 on success

**See Also:** SetTagRational Method, GetTagType Method

**Comments:** Scanner control tags are of different data types -- byte, short, long, ASCII (string), or rational. The **SetTagRational** method is used to set only rational tags. Refer to "Appendix B : Scanner Control Tags" on page 49 for details on the scanner control tags that are addressed through this method.

Either the **SetTagRational** or the **SetTagRational2** method may be used to set any rational tag. The difference between the two methods is that the true value must be calculated for **SetTagRational** while **SetTagRational2** accepts the numerator and denominator as separate values. For example, both of the following lines of code set the X-resolution to 300 DPI:

```
PixScan1.SetTagRational &H11A, 0, 65836
PixScan1.SetTagRational2 &H11A, 0, 300, 1
```

## ***SetTagString Method***

|                      |                                                                                                                                                                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Definition:</b>   | Sets the indexed tag element to the specified value. Must be a string tag.                                                                                                                                                                                                                                 |
| <b>Parameters:</b>   | Tag        ID of the tag to set<br>Value      Value to which to set tag                                                                                                                                                                                                                                    |
| <b>Syntax:</b>       | <code>PixScan1.SetTagString    <i>tagid</i>, "Letter Size"</code>                                                                                                                                                                                                                                          |
| <b>Return Value:</b> | 0 on failure<br>1 on success                                                                                                                                                                                                                                                                               |
| <b>See Also:</b>     | SetTagLong Method, SetTagRational Method, GetTagType Method                                                                                                                                                                                                                                                |
| <b>Comments:</b>     | Scanner control tags are of different data types -- byte, short, long, ASCII (string), or rational. The <b>SetTagString</b> method is used to set only ASCII tags. Refer to 'Appendix B : Scanner Control Tags' on page 49 for details on the scanner control tags that are addressed through this method. |

## ***SetupScanner Method***

|                      |                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Definition:</b>   | Opens a dialog box for manipulating the currently selected scanner's options.                                                                                                                                                                                                                                                                                                             |
| <b>Parameters:</b>   | None                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Syntax:</b>       | <code>PixScan1.SetupScanner</code>                                                                                                                                                                                                                                                                                                                                                        |
| <b>Return Value:</b> | 0 on failure or 'Cancel'<br>1 on success                                                                                                                                                                                                                                                                                                                                                  |
| <b>See Also:</b>     | SelectScanner Method, ConfigureScanner Method, 'Scanner Setup' on page 12                                                                                                                                                                                                                                                                                                                 |
| <b>Comments:</b>     | The dialog box displays standard setup information such as page size, brightness and contrast, and resolution. Those scanners that offer advanced features will generally have the 'Configure...' button enabled in this dialog. The advanced options dialog that is displayed when the 'Configure...' button is clicked may also be accessed through the <b>ConfigureScanner</b> method. |

## ***UnloadScanner Method***

|                      |                                                        |
|----------------------|--------------------------------------------------------|
| <b>Definition:</b>   | Unloads any scanner driver currently loaded in memory. |
| <b>Parameters:</b>   | None                                                   |
| <b>Syntax:</b>       | <code>PixScan1.UnloadScanner</code>                    |
| <b>Return Value:</b> | 0 on failure<br>1 on success                           |
| <b>See Also:</b>     | LoadScanner Method                                     |

**Comments:** This method does not change the value in the **Scanner** property. Executing this method will cause errors if the scanner driver is currently in use. Before unloading a scanner driver, ensure that all scanning processes and setup dialogs are closed.

### ***UseISISPipe Property***

**Definition:** If True, the PixScan control will attempt to pass any image data to a linked ImageBASIC control using Pixel Translations ISIS pipe architecture.

**Data Type:** Boolean

**Design Access:** Read/Write

**Runtime Access:** Read/Write

**Possible Values:** True (Default)  
False

**See Also:** "Linking Controls" on page 1

**Comments:** Although enabling this feature increases image transfer efficiency, the recipient control must also implement Pixel Translations technology. However, no error will be caused if the recipient control does not support this option.

### ***UseScannerCompression Property***

**Definition:** If True, the PixScan control will attempt to use any onboard compression that the scanner offers to improve throughput. If False, the scanner will output only uncompressed data to the host application.

**Data Type:** Boolean

**Design Access:** Read/Write

**Runtime Access:** Read/Write

**Possible Values:** True (Default)  
False

**Comments:** Some high performance scanners can output compressed image data to the host application. By default, ImageBASIC attempts to take advantage of this feature. In some circumstances, this option is not desired and can be disabled through the **UseScannerCompression** property.

**Note:** Enabling or disabling the compression of image data from the scanner does not affect the output of image data when a file is written from ImageBASIC.

### ***WillScan Event***

**Definition:** Occurs before each page is scanned.

**Parameters:** PageIndex The count of the page about to be scanned  
CancelScan If set to True during the event, scanning is halted

**See Also:** DidScan Event, ScanBatch Method, 'Scanning a Batch' on page 17

**Comments:** Any changes to the destination of the image about to be scanned should be performed in this event. For example, if the output file name for the page should be different, it must be set in this event. A related event, **DidScan**, occurs after each page is scanned and saved, if applicable.



# Appendix A : File Formats

---

## File and Compression Formats

Most of us do not have a lot of experience with image data. We may know that our favorite fax program produces PCX files, and that other programs produce files called TIFF files, but we may not know much beyond that. Many of the features in ImageBASIC will be much more easily understood if we present a short primer on how image data is captured, stored and maintained.

When a black and white image is captured at the scanner, it is held as raster data, or a stream of pixels that map together to form a picture. This data stream is relatively large; e.g., the pixels making up a bitonal 8.5" X 11" image scanned at 300 DPI (dots per inch) resolution will take up about 1 megabyte. The scanner usually sends this data either to the CPU or to a special board for compression.

As the data reaches the CPU there are numerous compression options. Each compression option presents different trade-offs with respect to compactness and decompression speed. For data that is being sent to the screen for display, we use a form of compression called Run Length Encoding (RLE) Run Length Encoding is a form of compressing the data that provides for very rapid decompression for quick display. RLE reduces the file size from 1 megabyte to about 250K, so it is somewhat more costly in terms of memory size than some other options.

By default, Diamond Head Software uses TIFF Group 4 compression in files and RLE in memory. Other options are available, however, and an overview of the various compression methods is given below.

## File Formats Supported by the PixScan Control

The following sections provide a brief description of the file types supported by this control. Included in the discussions are the types of color data that are generally included in these files and some limitations of each file type.

### ***BMP File Format***

A common Windows graphics format, BMP files can be read by almost any Windows-based image viewer. BMP files are paletted, and can represent up to 24-bit color. However, these files are not compressed and can therefore be very large when storing high color or high resolution images.

## ***CALS File Format***

CALS is a raster format for compressing bitonal image data. It was developed by the U.S. Department of Defense and is now in wide use throughout federal applications. The image data is either uncompressed or compressed using CCITT Group 4 algorithm. The image data is appended to a raster header block, very much like TIFF files. Although common in U.S. government document imaging applications, CALS is uncommon elsewhere.

## ***GIF File Format***

GIF has been popularized by years of use on CompuServe and other Internet resources. Typically grayscale or paletted, GIF images are widely supported and provide reasonable compression -- a typical GIF file will be 20% of the uncompressed data size. This file format can encode an image up to 64K X 64K pixels using from 1 to 8 bits of color.

GIF files use LZW compression and are, therefore, not recommended for use unless you have entered into a licensing agreement with UNISYS Corp., holders of rights to LZW.

## ***JPEG File Format***

JPEG is a compression method originally developed to compress photographs. As might be expected, most JPEG files are color or grayscale, not bitonal. JPEG is a "lossy" compression, meaning that some resolution and details are lost when an image undergoes compression. However, the loss of resolution is usually not visible and does not discourage the use of JPEG.

JPEG can encode an image of up to 24-bit color to a maximum size of 64K X 64K pixels. Many image files can be compressed to just 10% of their original size with very little net loss.

## ***PCX / DCX File Format***

PCX was developed and distributed by Microsoft and ZSoft and is common throughout all Windows installations. Image data is compressed using an RLE scheme, and is therefore quick but not markedly efficient, particularly when storing high color images. PCX can encode bitonal, 4-bit, 8-bit, or 24-bit color images up to 64,000 by 64,000 pixels. Typical compression for images or 16 colors or fewer is approximately 50%, while high color and complex images are compressed less.

DCX is a version of PCX that allows for the storage of multiple bitonal images in a single file.



## ***TIFF File Format***

TIFF was initially developed by Aldus Corporation to standardize the storage of bitonal images in document imaging and desktop publishing applications. TIFF is a versatile format composed of a header record listing details of the image followed by the compressed image data. Image data in a TIFF can be compressed using a number of different algorithms, and the exact compression scheme is recorded in the header to allow for easier decompression.

TIFF Group 4, the default format for ImageBASIC, compresses typical documents to approximately 5% of the original size. Group 4 is a bitonal standard and will not reliably store any grayscale or color images.

## **Compression Types**

Following are the most common data compression formats used in document imaging. Some additional information on compression may be found in the section "File Formats Supported by the PixScan Control" on page 43.

### ***CCITT Group 3 Compression***

Group 3 compression is a standard developed by CCITT, a standards organization committee responsible for the sanctioning of many file compression methods. This technique uses a combination of RLE, differential, and Huffman encoding.

### ***CCITT Group 4 Compression***

CCITT Group 4 compression is similar to Group 3 except that Group 3 limits its compression to one dimension, while Group 4 compresses both horizontally and vertically. Group 4 compression results in the smallest file size of all the options available in ImageBASIC, which is why it is the default compression method for file saving.

### ***LZW Compression***

LZW is a compression algorithm first developed in the late 1970's that is dictionary-based. LZW compression substitutes portions of image data that have been seen before with shorter strings stored in its dictionary.

The two file types compressed by this method that ImageBASIC supports are GIF and TIFF LZW. These files can be either bitonal or color, but LZW compresses paletted images best.

## ***Run Length Encoding (RLE)***

RLE (Run Length Encoding) is a simple compression algorithm that encodes a run of identical pixels as a count and the pixel value. This compression will generally result in a file size approximately one-quarter the original size. RLE compression is very fast but relatively inefficient; however, because of its speed advantages, it is used as the compression method for images held in memory by ImageBASIC and also in BMP files.

## ***Uncompressed Image Files***

Uncompressed images are simply that -- no compression has been applied to the raw image data. Therefore, the image files tend to be very large. For example, the raw data to describe a bitonal letter size image at 300 DPI is about 1 MB. Adding to the area of the image or to the level of color can drastically increase the file size and the time required to transfer and display it.

---

## **Color Limitations of File Formats**

In the following tables, the entries in the top row of each table indicate the color format of an image file. The three numbers indicate, respectively, BitsPerSample, SamplesPerPixel, and PhotometricInterpretation. The left-most column indicates the type of file format. By finding the intersection of the file type and color format, you can find the most appropriate, fully functional combination for your needs.

**BitsPerSample(BPS)** indicates how many different values are allowed for each sample (see SamplesPerPixel) that defines a pixel's color. The only valid values are 1, 4, and 8.

**SamplesPerPixel(SPP)** indicates how many individual values are used to define a pixel's color. The only valid values are 1 and 3. A value of 1 indicates that this image is not RGB, leaving bitonal, grayscale, and paletted as possibilities. BitsPerSample and PhotometricInterpretation will indicate which of these possible formats is actually used.

**PhotometricInterpretation(PI)** indicates the interpretation of color values for display. The valid values of PhotometricInterpretation are as follows:

- |   |         |                                                                                                      |
|---|---------|------------------------------------------------------------------------------------------------------|
| 0 | White 0 | Typical bitonal/gray interpretation in which 0 indicates white and higher numbers are darker shades. |
| 1 | White 1 | The inverse of White 0, in which 0 is interpreted as black and higher numbers are lighter shades.    |

- 2 RGB Red, Green, and Blue values of each pixel are specified; this setting requires a value of 3 for SamplesPerPixel
- 3 Paletted An array is constructed with a color assigned to each value in than array. This value is generally only applied to color images and is limited to few values than RGB.

The key to the results codes in the tables is as follows:

- Y The combination file type and color definition is possible based on the specifications for both file type and compression format.
- N This combination is not legal based on the specifications for the file type and or compression format.

### Bitonal and Grayscale Image File Formats

|                       | Binary<br>BPS 1<br>SPP 1<br>PI 0/1 | 16-level gray<br>BPS 4<br>SPP 1<br>PI 0/1 | 256-level gray<br>BPS 8<br>SPP 1<br>PI 0/1 |
|-----------------------|------------------------------------|-------------------------------------------|--------------------------------------------|
| TIFF Group 4          | Y                                  | N                                         | N                                          |
| TIFF Group 3          | Y                                  | N                                         | N                                          |
| TIFF Group 3 Modified | Y                                  | N                                         | N                                          |
| TIFF Packed           | Y                                  | Y                                         | Y                                          |
| TIFF Uncompressed     | Y                                  | Y                                         | Y                                          |
| TIFF LZW              | Y                                  | Y                                         | Y                                          |
| CALS                  | Y                                  | N                                         | N                                          |
| PCX                   | Y                                  | N                                         | Y                                          |
| DCX                   | Y                                  | N                                         | N                                          |
| PDA Uncompressed      | Y                                  | N                                         | N                                          |
| PDA Group 4           | Y                                  | N                                         | N                                          |
| PDA Group 3           | Y                                  | N                                         | N                                          |
| GIF                   | Y                                  | N                                         | Y                                          |
| BMP                   | Y                                  | N                                         | Y                                          |
| JPEG                  | N                                  | N                                         | N                                          |

## Paletted and RGB Image File Formats

|                       | 16-color palette<br>BPS 4<br>SPP 1<br>PI 3 | 256-color palette<br>BPS 8<br>SPP 1<br>PI 3 | 24-bit RGB<br>BPS 8<br>SPP 3<br>PI 2 |
|-----------------------|--------------------------------------------|---------------------------------------------|--------------------------------------|
| TIFF Group 4          | N                                          | N                                           | N                                    |
| TIFF Group 3          | N                                          | N                                           | N                                    |
| TIFF Group 3 Modified | N                                          | N                                           | N                                    |
| TIFF Packed           | Y                                          | Y                                           | Y                                    |
| TIFF Uncompressed     | Y                                          | Y                                           | Y                                    |
| TIFF LZW              | Y                                          | Y                                           | N                                    |
| CALS                  | N                                          | N                                           | N                                    |
| PCX                   | N                                          | Y                                           | Y                                    |
| DCX                   | N                                          | N                                           | N                                    |
| PDA Uncompressed      | N                                          | N                                           | N                                    |
| PDA Group 4           | N                                          | N                                           | N                                    |
| PDA Group 3           | N                                          | N                                           | N                                    |
| GIF                   | Y                                          | Y                                           | N                                    |
| BMP                   | Y                                          | Y                                           | N                                    |
| JPEG                  | N                                          | N                                           | Y                                    |

# Appendix B : Scanner Control Tags

---

## Overview of Tag Support

This document includes a great many tags that are available only for select scanner models and will not be available on the majority of scanners. In addition, even those tags that are supported by a given scanner may not offer all the options listed with each tag. This is due to hardware design and limitations. Other factors contributing to which tags are available are the optional features available on some scanners. For example, the Ricoh IS-520 offers, among others, options for InkJet endorsing, electronic endorsing, job separation, and high resolution scanning, each of which may be independently added to any given scanner. The tags to support each of these options will not be recognized by the scanner driver unless that hardware option is available.

In addition to the limitations imposed by scanner model, some tags will be available only under certain circumstances. For example, it is common for scanner to make TAG\_CONTRAST a read-only value when bitonal scanning has been selected. If gray scale or color scanning is later chosen, TAG\_CONTRAST can be set.

All of the tags documented here are used to specify or read the parameters of one or more scanners. Those tags that are used by relatively few scanners (and in some cases, only one model) are identified as such whenever possible.

The Index lists all tags alphabetically to aid in the location of one if its name is already known. The Index also includes some functional and scanner model groupings to help developers find the most useful tags as quickly as possible.

## Scanner Tag Data Types

Scanner tags are one of five data types -- byte, short, long, string, or rational. Because of these different data types, different methods are available in the Pixel Scanning control to access each type of tag.

For each type of tag, ImageBASIC includes several methods to read and set the tag. All of these methods are described in the main document.

The following methods apply to all tags, regardless of the tag's data type:

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetTagChoiceCount   | <p>Returns a long number totaling the number of legal values for the specified tag. This method will return the number of legal values for a tag even if the tag is a range type (see <b>GetTagChoiceFlags</b> method).</p> <p>To get the valid values for any tag, use one of the following methods, depending upon the data type of the tag:</p> <p>GetTagChoiceLong</p> <p>GetTagChoiceString</p> <p>GetTagChoiceRational</p> |
| GetTagChoiceFlags   | <p>Returns a long value of the type of tag -- range or list. A range type tag has values represented internally as a high value, a low value, and a step value. A list type tag has values represented as a fix list of the valid options.</p>                                                                                                                                                                                   |
| GetTagLength        | <p>Reports the number of indexed elements in a tag. A few tags maintain an array of values. This method queries the tag to find out how many elements are in the array. However, most tags hold only one value, and for these tags, this method will return a value of 1 (one).</p> <p>If an ASCII tag is queried, the number of characters in the current setting will be returned.</p>                                         |
| GetTagLengthDefault | <p>Reports the number of indexed elements for the tag's default value. A few tags maintain an array of values. This method queries the tag to find out how many elements are in the array. However, most tags hold only one value, and for these tags, this method will return a value of 1 (one).</p> <p>If an ASCII tag is queried, the number of characters in the default setting will be returned.</p>                      |

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetTagType    | <p>Reports the data type of the specified tag. The return value of this method will correspond to one of the following values:</p> <ol style="list-style-type: none"> <li>1      Byte *</li> <li>2      ASCII</li> <li>3      Short *</li> <li>4      Long *</li> <li>5      Rational</li> </ol> <p>* For the purposes of choosing the proper method to read and write tag values, all Byte, Short, and Long tags are considered Long.</p> |
| SetTagDefault | <p>Sets the specified tag to its default value. This method will set any data type of tag -- long, rational, string -- to its default value. If this method is called specifying a tag ID of 0 (zero), all tags will be set to defaults.</p>                                                                                                                                                                                               |

The following methods are used to read and write to the scanner control tags based upon the data type of the tag.

All tags that hold byte, short or long numbers are addressed through the following methods:

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetTagLong        | <p>Reads the current value of the specified long tag and returns a long number indicating the current value of the specified tag element</p> <p>Most tags hold only one value at a time and will return a meaningful value for only Index = 0. This method can only read the values for tags that are short (two byte) or long (four byte) data types. Refer to "GetTagType Method" on page 29 for determining a tag's data type.</p> |
| GetTagLongDefault | <p>Reports the default value for the specified element of a tag and returns a long number indicating the default value of the specified index. This method will read only tags that maintain short or long data type values.</p> <p>Most tags hold only one value at a time, because they have only one index element. Therefore, for most tags, this method will be called with the Index parameter set to 0 (zero).</p>             |
| GetTagChoiceLong  | <p>Reports the value of one option in a tag's list of legal values. Applies only to tags accepting long data type</p>                                                                                                                                                                                                                                                                                                                 |

values. All numerical tags, whether they are enumerated (list type) or have a range of values (range type), maintain a list of all of their legal values. Each value is assigned an index in that list.

This method reads the value assigned to an index, allowing the application to find all of the legal values for any tag that accepts long values.

For example, the following Visual Basic code reads all of the values for TAG\_PHOTOMETRICINTERPRETATION (106 hex) that are supported on the currently selected scanner and displays them in a list box.

```
count = PixScan1.GetTagChoiceCount &H106
For i = 0 to (count - 1)
    Opt = PixScan.GetTagChoiceLong &H106, i
    List1.AddItem Opt
Next i
```

#### SetTagLong

Sets the indexed tag element to the specified long tag. Scanner control tags are of different data types -- byte, short, long, ASCII (string), or rational. The data type of each tag is given in the description of each tag or may be found using the **GetTagType** method. The **SetTagLong** method is used to set byte, short, and long tags.

All tags that hold string values are addressed through the following methods:

**GetTagChoiceString** Reads the legal value for a tag that has been assigned the specified index value. Valid only for string tags.

Each scanner driver maintains a list of all valid options for every supported tag. This method allows the user to find the legal values for the tag.

**GetTagString** Reads the current value of the specified string tag. All of the currently supported string tags have only one indexed element, so this method will always be called with an Index of 0 (zero). The most commonly used string tag is TAG\_PAGESIZE.

**GetTagStringDefault** Returns the default value of the specified string tag.

**SetTagString** Sets the indexed tag element to the specified value. Must be a string tag.

All tags that maintain rational values are addressed through the following methods:



|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetTagChoiceRational  | <p>Reports the value of one legal option in a tag's list of legal values for all tags accepting rational values. Returns a double (4 byte) numerical value of the option at the specified index. This method may be used only with rational value type tags.</p> <p>All numerical tags, whether they are enumerated (list type) or have a range of values (range type), maintain a list of all of their legal values. Each legal value is assigned an index in that list. This method reads the value assigned to an index, allowing the application to find all of the legal values for any tag that accepts rational values. TAG_XRESOLUTION and TAG_YRESOLUTION are the most commonly used rational tags.</p> |
| GetTagRational        | <p>Reads the current value of the specified rational tag. This method will return the current value for rational tags only. The only two commonly used rational tags are TAG_XRESOLUTION and TAG_YRESOLUTION.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| GetTagRationalDefault | <p>Reports the default value for the specified rational tag and returns a double (signed 8 byte) number indicating the default for the specified element in the tag. All of the currently supported rational tags have only one indexed element, so this method will always be called with an Index of 0 (zero). The only rational tags that are commonly used are TAG_XRESOLUTION and TAG_YRESOLUTION.</p>                                                                                                                                                                                                                                                                                                      |
| SetTagRational        | <p>Sets the indexed tag element to the specified value. Must be a rational tag.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

### ***Interpreting Rational Tag Values***

The values of typical scan tags (those that maintain integer values) are interpreted either directly as they appear or according to an enumerated list, shown with each tag in the descriptions below. Rational tags hold values that must be transformed to be easily recognizable.

To interpret a rational tag's value, that value must be considered as a binary number equivalent to the decimal number that is returned. As noted above, rational tags are of the long data type, which is a four-byte number. This number must be converted to a binary format, and the upper and lower two bytes separated.

In some cases, the two parts of the binary number will be converted to a fraction, and in others, each half is considered independently. An example of each type of

interpretation is shown below, and each rational tag in this documentation provides details on its own peculiarities.

### **Rational Tag Values as Fractions**

The most commonly used rational tags specify resolution -- TAG\_XRESOLUTION and TAG\_YRESOLUTION. If TAG\_XRESOLUTION is queried using the GetTagRational method, a long integer will be returned. If the scanner's resolution is set to 300 DPI, the value will be 65836, or some multiple of this number.

Converted to a four-byte binary number, 65836 (dec) is 0000 0000 0000 0001 0000 0001 0010 1100.

The lower two bytes of this are 0000 0001 0010 1100 and the upper two bytes are 0000 0000 0000 0001.

Place these numbers in a fraction 0000 0000 1001 0110 / 0000 0000 0000 0001 which in decimal format is 300 / 1 or 300 DPI.

For this type of rational value, another formula may be used to determine the final value. This formula is used for TAG\_XRESOLUTION and TAG\_YRESOLUTION, as well as a few others. Each tag that may be interpreted using this formula will say so in the tag's description. The formula below will emulate the process of converting to binary, separating the upper and lower halves, and forming a fraction from the resulting numbers:

TagVal = PixScan1.GetTagRational &H11A, 0

$$DPI = \frac{(TagVal) MOD 2^{16}}{INT(TagVal / 2^{16})}$$

### **Rational Tag Values as Independent Numbers**

Those rational tags that encode two independent numbers generally indicate the X and Y values in a positional array (x, y). One such tag is TAG\_WHITEPOINT. The value of this tag refers to a position on the 1931 CIT chromaticity diagram. The upper two bytes of the value of this tag correspond to the X value, and the lower two bytes correspond to the Y value. To find each of these values, the number must be converted to binary, split, and each half reconverted to decimal. This process is not easily accomplished in Visual Basic code, but these tags are rarely if ever used.

## **Ordering the Setting of Tags**

The order in which tags are set may be important. The setting of one tag may change the range of values available in other tags or may reset them to default values. Two common tags that reflect this behavior are page size and contrast.

Page size will usually be reset when resolution is changes, unless the Image Display Window property PageSize is set to Manual, in which case the developer must ensure that the region specified is of a legal size. Most scanners will only allow contrast settings to be made when outputting either gray scaled or dither data.

The following list shows the preferred order for setting these tags, and although other arrangements are possible, this one has been demonstrated to be applicable to most scanners. Most of these tags, with the exception of those marked with an asterisk (\*), are available for most scanners.

TAG\_SAMPLESPERPIXEL  
TAG\_BITSPERSAMPLE  
TAG\_PHOTOMETRICINTERPRETATION\*  
TAG\_DITHER  
TAG\_XRESOLUTION  
TAG\_YRESOLUTION  
TAG\_PAGESIZE  
TAG\_XPOSITION  
TAG\_YPOSITION  
TAG\_IMAGELENGTH  
TAG\_IMAGEWIDTH  
TAG\_COMPRESSION    Commonly available, often read-only and uncompressed  
TAG\_PLANARCONFIGURATION\*    For color scanning only  
TAG\_CONTRAST  
TAG\_BRIGHTNESS  
TAG\_GAMMA\*  
TAG\_OUTLINE\*  
TAG\_EMPHASIS\*  
TAG\_MIXEDSCAN\*

---

## Reference to the Scanner Control Tags

### Resolution and Page Size Scanner Control Tags

The following tags specify features of the scanned image such as the resolution of the final image, the number of colors in the image, and the area of the page that is to be scanned.

## ***TAG\_XRESOLUTION***

ID Number (hex): 11a

Data Type: Rational

Specifies the horizontal resolution that will be used for a scan. Because this is a rational tag, it is not set to simple DPI values, but instead its value is the decimal equivalent of a 4 byte binary number in which the top 2 (two) bytes are the denominator and the bottom 2 (two) bytes are the numerator.

To simplify the setting of this tag and TAG\_YRESOLUTION, the following decimal values may be used. Of course, only those resolutions supported by the scanner in question can be successfully set.

65566 => 30 DPI

65584 => 48 DPI

65596 => 60 DPI

65611 => 75 DPI

65636 => 100 DPI

65656 => 120 DPI

65686 => 150 DPI

65736 => 200 DPI

65836 => 300 DPI

66136 => 600 DPI

If your scanner returns different values than those shown above when this tag is queried, the following formula may be used to convert these decimal return values to DPI, or the return value will be a multiple of one of the values shown above:

`TagVal = PixScan1.GetTagRational (&H11A, 0)`

$$DPI = \frac{(TagVal) MOD 2^{16}}{INT(TagVal / 2^{16})}$$

Changing the value of this tag or TAG\_YRESOLUTION often resets TAG\_PAGESIZE, X and Y Offset, and image width and image length tags to the defaults. Therefore, care should be taken to set these tag values in the proper order.

**Note:** Many scanners will accept the exact DPI value you wish to set when defining the scanning resolution, as shown here to specify 300 DPI:

`PixScan1.SetTagRational &H11A, 0, 300`

## ***TAG\_YRESOLUTION***

ID Number (hex): 11b

Data Type: Rational

Specifies the vertical resolution. When queried with the **GetTagRational** method, this tag returns decimal values as TAG\_XRESOLUTION, above. The returned values for each DPI and the conversion formula are the same. Just as with TAG\_XRESOLUTION, this tag is set with the **SetTagRational** method, and the example below shows vertical resolution being set to 100 DPI:

```
PixScan1.SetTagRational &H11B, 0, 100
```

**Note:** Many scanners will accept the exact DPI value you wish to set when defining the scanning resolution.

## ***TAG\_RESUNIT***

ID Number (hex): 128

Data Type: Long

TAG\_RESUNIT specifies the unit used by TAG\_XRESOLUTION and TAG\_YRESOLUTION. The default value is *2--Inches*, as most applications assume this as the standard unit. If using a value other than inches, carefully test the receiving application for compatibility. If set to *None*, the values in the resolution tags are taken as an aspect ratio.

- 1 None
- 2 Dots per Inch
- 3 Dots per Centimeter

To change the resolution unit to dots per centimeter, set this tag as shown below:

```
PixScan1.SetTagLong &H128, 0, 3
```

Where *&H128* is the tag ID number, *0* is the index into the tag (must be zero for this tag), and *3* is the value to assign.

## ***TAG\_PAGESIZE***

ID Number (hex): 50e

Data Type: String

TAG\_PAGESIZE may be used to specify the page size setting of the scanner. This tag must be set to an exact string value, which varies by scanner model. Appendix A lists the proper strings for some scanners. If your scanner is not listed, the valid options may be found by calling the **SetupScanner** dialog box and recording the exact strings shown in the page size list. The current value of this tag may be found by querying the **GetTagString** method, which will return a string data type:

```
ret$ = PixScan1.GetTagString &h50E, 0
```

TAG\_PAGESIZE is affected by the values in several other properties and changing those values will reset page size. Also, changing page size will alter yet more tag values. Refer to 'Ordering the Setting of Tags' on page 54 for a list of the preferred order.

Once a list of the proper strings for this tag are found, the **SetTagAscii** method may be used to select a page size. For example, to select letter size pages, many scanners require the following setting:

```
PixScan1.SetTagString &H50E, "Letter Size - 8.5 X 11 in"
```

A few scanners do not support TAG\_PAGESIZE. For these scanners, the page size is defined by first setting TAG\_XRESOLUTION and TAG\_YRESOLUTION, followed by setting TAG\_IMAGEWIDTH and TAG\_IMAGELENGTH.

## **TAG\_DETECTPAGESIZE**

ID Number (hex): 57b

Data Type: Long

This tag specifies whether page size detection will occur on those scanners capable of doing so. When this tag is set to *1--On*, TAG\_XPOSITION, TAG\_YPOSITION, TAG\_IMAGELENGTH, and TAG\_IMAGEHEIGHT are reset to their defaults.

0 Off

1 On

Bell & Howell SCSI -- Image width is set to the width of the currently selected page size plus two inches, if possible. Image height is set to current page size plus 0.90 inches. If an ADF is in use, the driver re-centers the X-Offset as if the left paper guide has moved an inch to the left. If the area searched is larger than 1.755 KB, the scanner will stop scan ahead and performance will be notably degraded.

## **TAG\_XPOSITION**

ID Number (hex): 11e

Data Type: Long

This tag specifies the distance from the left edge of the page to the beginning of the scan. The unit used for this tag is generally pixels, but TAG\_RESUNIT may alter it for some scanners. Changing any of the following tags, if they are available on the scanner in question, will usually reset TAG\_XPOSITION:

TAG\_XRESOLUTION TAG\_WINDOW

TAG\_PAGESIZE TAG\_SUBWINDOW

## ***TAG\_YPOSITION***

ID Number (hex): 11f

Data Type: Long

This tag specifies the distance from the top edge of the page to the beginning of the scan. The unit used for this tag is generally pixels, but TAG\_RESUNIT may alter it for some scanners. Changing any of the following tags, if they are available on the scanner in question, will usually reset TAG\_YPOSITION:

TAG\_XRESOLUTION TAG\_WINDOW

TAG\_YRESOLUTION TAG\_SUBWINDOW

TAG\_PAGESIZE

## ***TAG\_IMAGELENGTH***

ID Number (hex): 101

Data Type: Long

The scan area may be changed from the page size specified in TAG\_PAGESIZE using TAG\_XPOSITION, TAG\_YPOSITION, TAG\_IMAGELENGTH, and TAG\_IMAGEWIDTH. TAG\_IMAGELENGTH specifies the length in pixels of the area to be scanned. Care must be taken to ensure that the area specified by TAG\_IMAGELENGTH and TAG\_YPOSITION does not exceed the readable area of the scanner's platen. Also, changing TAG\_YRESOLUTION or TAG\_PAGESIZE will change the area described by TAG\_IMAGELENGTH.

## ***TAG\_IMAGEWIDTH***

ID Number (hex): 100

Data Type: Long

Like TAG\_IMAGELENGTH, TAG\_IMAGEWIDTH is used to specify the size of the area to be scanned. Care must be taken to ensure that the area specified between TAG\_XPOSITION and TAG\_IMAGEWIDTH does not exceed the scanner's physical limitations, remembering that a change in TAG\_XRESOLUTION will alter the size of the defined area.

## ***TAG\_SCANORIENTATION***

ID Number (hex): 113

Data Type: Short

TAG\_SCANORIENTATION specifies the expected orientation of page fed into the scanner so that output images may be automatically made portrait.

- 1 Portrait
- 2 Landscape

## **Contrast and Brightness Scanner Control Tags**

Just as brightness and contrast may be set for gray scaled and dithered images, and will change the appearance of the output image, each of these settings may be made for each primary color. By individually setting these colors, the user may correct color interpretation of distorted images or may create unique effects in the output image file.

In general, brightness and contrast function as follows. As an image is scanned, the apparent color of each pixel is recorded. Setting brightness to a level above normal will cause all pixels in the output image to be equally brighter than they were in the original page. Likewise, setting brightness to a lower value will cause the output image pixels to all be recorded as darker than they were in the original. Contrast functions with brightness by increasing or decreasing the total range of colors that will be present in the output file. A contrast value above normal will extend the range of colors in the output file. This result in the darker shades in the original image being recorded as even darker in the output image, and the brighter shades in the original being recorded as even brighter. Setting contrast to a low value reduces the overall range of color in the output image, relative to the original. Low contrast values cause darker shades to be recorded as brighter, and the brighter shades to be recorded as darker.

Using the following tags, the brightness and contrast of each primary color may be individually manipulated. However, just as with TAG\_CONTRAST and TAG\_BRIGHTNESS, the range of valid values for these tags is entirely scanner



dependent. Also keep in mind that these tags will be disabled until a color scanning setup has been selected.

### ***TAG\_BRIGHTNESS***

ID Number (hex): 502

Data Type: Long

TAG\_BRIGHTNESS specifies the brightness level of a scan. The following values are common, and many scanners accept exact settings, too. The valid range of settings for any one scanner may be found by looking in the **SetupScanner** dialog box.

- 3 Auto -- Scanner dynamically selects brightness according to conditions
- 2 Darken -- Darkens the image relative to the current setting
- 1 Normal -- Sets to the middle of the available range
- 0 Lighten -- Lighten the image relative to the current setting

### ***TAG\_CONTRAST***

ID Number (hex): 501

Data Type: Long

TAG\_CONTRAST specifies the relative level of contrast applied to each image as it is scanned. The exact range of values varies widely by scanner model, but 0 is commonly an automatic setting. The valid range of settings for any one scanner may be found by looking in the **SetupScanner** dialog box. Contrast may usually be set only when image output is grayscale or dithered.

For Bell & Howell scanners with either the 3-function or 5-function ACE boards, the scanner is instructed how to enhance the scan by setting TAG\_CONTRAST.

### ***TAG\_GREENCONTRAST***

ID Number (hex): 5d1

Data Type: Long

This tag specifies the contrast setting for the green component in all pixels of the output image relative to the original.

### ***TAG\_REDCONTRAST***

ID Number (hex): 5d2

Data Type: Long

This tag specifies the contrast setting for the blue component in all pixels of the output image relative to the original.

## ***TAG\_REDBRIGHTNESS***

ID Number (hex): 5d3

Data Type: Long

This tag specifies the brightness setting for the red component in all pixels of the output image relative to the original.

## ***TAG\_GREENBRIGHTNESS***

ID Number (hex): 5d4

Data Type: Long

This tag specifies the brightness setting for the green component in all pixels of the output image relative to the original.

## ***TAG\_BLUEBRIGHTNESS***

ID Number (hex): 5d5

Data Type: Long

This tag specifies the brightness setting for the blue component in all pixels of the output image relative to the original.

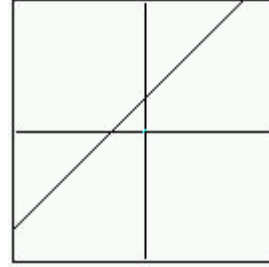
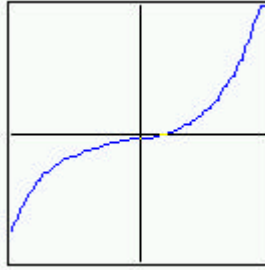
## **Gamma Correction**

Gamma correction is similar to the specification of a brightness setting, except that colors at different points in the spectrum may be interpreted differently. That is, changing the brightness value will cause all output colors to be brighter or darker than the original. This change will be exactly the same for all colors, regardless of their position in the color spectrum. This processing is represented by a straight line that indicates that all colors are brightened or darkened by an equal amount (see the table below).

Gamma correction, on the other hand, allows each point in the color spectrum to be processed differently, perhaps by lightening the darker shades, not changing mid-range tones, and darkening the upper spectrum. This would result in more nearly uniform colors throughout the output image. A diagram representing this type of correction is shown below.

**Graphic Representation  
of a Gamma Curve**

**Graphic Representation  
of a Brightness Setting**



## ***TAG\_GAMMA***

ID Number (hex): 50f

Data Type: Short

TAG\_GAMMA allows certain preset gamma curves to be selected. It is unlikely that any one scanner supports all the options shown below, so the developer will have to either experiment or look in the **SetupScanner** dialog box to find which options are available. Some scanners allow the creation of custom gamma curves through TAG\_GAMMA\_LENGTH $x$ , where  $x$  is an integer between one and six.

- 0 AUTO
- 3 BOTTOMCLIP
- 2 1TO1INVERT
- 1 SCURVE
- 1 SHARP\_61
- 2 SHARP\_67
- 3 SHARP\_75
- 4 SHARP\_87
- 5 1TO1
- 6 BLUR\_1\_20
- 7 BLUR\_1\_45
- 8 BLUR\_1\_78
- 9 BLUR\_2\_20

## **Color Definition Scanner Control Tags**

### ***TAG\_BITSPERSAMPLE***

ID Number (hex): 102

Data Type: Long

TAG\_BITSPERSAMPLE defines the number of bits used to define each sample for a pixel. The following values are valid for this tag:

- 1 Binary image

- 4 4-bit gray scale, 4-bit paletted, or 12-bit RGB image
- 8 8-bit gray scale, 8-bit paletted, or 24-bit RGB image

For more details on color definition, see TAG\_SAMPLESPERPIXEL.

## ***TAG\_SAMPLESPERPIXEL***

ID Number (hex): 115

Data Type: Long

Considered in conjunction with TAG\_BITSPERSAMPLE, TAG\_SAMPLESPERPIXEL defines the number of colors or shades of gray that may be present in the output image. TAG\_PHOTOMETRICINTERPRETATION specifies whether the image will be bitonal/grayscale, paletted, or RGB. TAG\_SAMPLESPERPIXEL defines the number of color samples used for each pixel. TAG\_BITSPERSAMPLE defines the number of different intensities within that sample.

For example, if TAG\_SAMPLESPERPIXEL is 1, and TAG\_BITSPERSAMPLE is 4, one sample will define each pixel, and 4 bits will be used to define that sample, for a maximum range of 16 colors. If TAG\_PHOTOMETRICINTERPRETATION is 0 or 1, the image will be grayscale, and if it is 3, the image will be paletted.

Likewise, if TAG\_SAMPLESPERPIXEL is set to 3, and TAG\_BITSPERSAMPLE is set to 8, three samples will define each pixel, and 8 bits will be used to define that sample, for a maximum range of slightly more than 65 million colors. This combination is generally used only with TAG\_PHOTOMETRICINTERPRETATION set to 2, or RGB.

## ***TAG\_PHOTOMETRICINTERPRETATION***

ID Number (hex): 106

Data Type: Long

TAG\_PHOTOMETRICINTERPRETATION (PMI) specifies the output color format of images. White0 and White1 indicate either bitonal or grayscale images (depending upon the values in TAG\_BITSPERSAMPLE (BPS) and TAG\_SAMPLESPERPIXEL (SPP)). 2--*RGB* indicates an RGB image, which requires a SPP of 3 and a BPS of 4 or 8. 3--*Paletted* requires a SPP of 1 and a BPS of 4 or 8.

- 0 White0
- 1 White1
- 2 RGB
- 3 Paletted

## ***TAG\_DITHER***

ID Number (hex): 500

Data Type: String

TAG\_DITHER specifies the type of dithering done by the scanner as it is processing each image. The acceptable values for this tag vary by scanner model.

# **Image Data Storage Scanner Control Tags**

## ***TAG\_COMPRESSION***

ID Number (hex): 103

Data Type: Long

As image data is sent from the scanner to an ImageBASIC application, that data may be compressed to reduce transfer time and optimize scanner performance. If a particular scanner model is capable of outputting compressed data, and those that are capable usually default to do so, TAG\_COMPRESSION may be used to specify the type of data compression that is used.

- 1 None -- No compression is used -- However, the pixel information is packed into bytes as tightly as possible. Each row of bitmap information is padded with arbitrary data to fill out to a byte boundary, and each line starts on a byte boundary.
- 2 CCITT Group 3 1-Dimensional Modified Huffman run length encoding --
- 3 Fax compatible Group 3 -- Each strip begins on a byte boundary and the data is stored as bytes, not words. Terminating characters are used so that fax communication is possible, and because of the terminating

characters, rows that are not the first row of a strip are not required to start on a byte boundary.

- 4 CCITT Group 4 (Fax compatible) -- Each strip must begin on a byte boundary, and the data is stored as bytes, not words. Rows that do not begin a strip are not required to start on a byte boundary because terminating characters are used.
- 5 LZW Compression for grayscale, paletted, and RGB images. LZW is a dictionary-based encoding algorithm which builds a translation table of data occurring in the uncompressed data stream. Patterns in the data stream are matched to entries in the table.
- 6 32771 -- Uncompressed data, much like TAG\_COMPRESSION = None, but each line starts on an even byte boundary.

### ***TAG\_GROUP3OPTIONS***

ID Number (hex): 124

Data Type: Long

When outputting Group3 format images (see TAG\_COMPRESSION), a scanner may be able to output certain variations of the Group 3 format.

TAG\_GROUP3OPTIONS specifies which of these options is used. If TAG\_COMPRESSION is set to other than Group 3, the value of TAG\_GROUP3OPTIONS will be ignored.

- 0 None -- Standard 1-dimensional encoding, the default and most widely acceptable option.
- 1 2-Dimensional encoding. If more than one strip is specified, each strip must begin with a 1-dimensional line. In other words, TAG\_ROWSPERSTRIP should be a multiple of TAG\_KFACTOR.
- 2 Uncompressed Group 3 encoding.
- 4 Fill bits will be added as necessary before End Of Line codes so that each EOL ends on a byte boundary, thus ensuring an EOL sequence of one byte preceded by a zero nibble: xxxx-0000 0000-0001.

## ***TAG\_GROUP4OPTIONS***

ID Number (hex): 125

Data Type: Long

When a scanner is outputting image data in a Group 4 compression format (see TAG\_COMPRESSION), certain options to this format may be selected by setting TAG\_GROUP4OPTIONS. It is advisable to not alter this tag value.

- 0 Default -- Most common and most broadly applicable option. ImageBASIC expects this value when accepting Group 4 compressed images.
- 2 Uncompressed -- Uncompressed Group 4 encoding.

## ***TAG\_FILLORDER***

ID Number (hex): 10a

Data Type: Long

The use of this tag is no longer recommended by the CCITT. Developers are encouraged to support the default value of 1.

- 1 MSBL -- Most Significant Bit Left -- As each pixel is placed into a byte according to the fill order, it is placed into a byte from left to right (from most significant to least significant position). The default and recommended value.
- 2 MSBR -- Most Significant Bit Right -- As each pixel is placed into a byte according to the fill order, it is placed from right to left (from the least significant to the most significant position).

## ***TAG\_PLANARCONFIGURATION***

ID Number (hex): 11c

Data Type: Long

This tag describes how color data is organized, and is only relevant when SamplesPerPixel is 3. Bitmaps that represent color data can be organized either as color by pixel or color by plane. If TAG\_PLANARCONFIGURATION is set to 1 then a 3 SamplesPerPixel red, green blue image would have pixels stored R1, G1, B1, R2, G2, B2, R3, G3, B3...etc. If the Planar Configuration is set to 2, then each color plane is stored contiguously, or R1, R2, R3, ... G1, G2, G3, ... B1, B2, B3...etc. The value of TAG\_PHOTOMETRICINTERPRETATION must be taken into account when interpreting this tag.

- 0 LINE -- Default
- 1 LINE\_PACK
- 2 PAGE
- 5000 LINE\_UNPACK

## ***TAG\_STRIPOFFSETS***

ID Number (hex): 111

Data Type: Long

TAG\_STRIPOFFSETS is a pointer to the start of a strip of image data in the TIFF file. In the case of a single strip image, this value fits in the four bytes tag field. In the case of a multiple strip image, the necessary range of values is greater, and the 44-byte field contains an offset that points to an array of pointers, one for each strip. This tag should almost never be altered by an ImageBASIC developer.

## ***TAG\_ROWSPERSTRIP***

ID Number (hex): 116

Data Type: Long

TAG\_ROWSPERSTRIP indicates the number of rows that are present on each strip. This requires that all strips except the last one represent the same number of strips. If there is only one strip in the image, then TAG\_ROWSPERSTRIP is set at its maximum value of  $2^{16}-1$  or  $2^{32}-1$ , depending on the data type.

The default value for this tag is  $2^{12}-1$ , and should not be changed except to specify the actual number of rows in the image. For ImageBASIC users, setting this tag is handled by the application, and the user should not change its value.

## ***TAG\_STRIPBYTECOUNTS***

ID Number (hex): 117

Data Type: Long

Each TAG\_STRIPOFFSETS value has associated with it a TAG\_STRIPBYTECOUNTS value. This information is used to verify that the correct number of bytes are read when doing decompression and may also be used to allocate memory for the strip of compressed data when performing decompression. The values for this tag are determined during data compression -- after all the rows in each strip have been compressed, the size of the strip becomes the value used for TAG\_STRIPBYTECOUNTS. This is a read-only tag.

## ***TAG\_ORIENTATION***

ID Number (hex): 112

Data Type: Long

TAG\_ORIENTATION specifies the output rotation from the scanner. These rotations are relative to the input, and are not based on any determination of actual input orientation. In other words, the rotation specified here assumes that pages are put in the scanner portrait. If pages are fed into the scanner landscape and TAG\_ORIENTATION is set to *1--Portrait*, the output images will still be



landscape, just like the originals. Likewise, with pages fed into the scanner landscape and TAG\_ORIENTATION set to 2--*Landscape*, the output image will be upside down.

- 1 0 degrees (portrait mode) Default
- 2 90 degrees (landscape mode)
- 3 180 degrees
- 4 270 degrees
- 5 0 degrees & flipped horizontally
- 6 90 degrees & flipped horizontally
- 7 180 degrees & flipped horizontally
- 8 270 degrees & flipped horizontally

## Image Data Interpretation Scanner Control Tags

### ***TAG\_COLORRESPONSEUNIT***

ID Number (hex): 12c

Data Type: Long

Because color response curves, as defined in TAG\_COLORRESPONSECURVES, accepts values between 0 and 2 including fractional values (e.g., 1.2 or 0.4), but the tag may only be set to integer values, TAG\_COLORRESPONSEUNIT specifies a constants by which to multiply that number so that integer values are produced for the tag. For example, if TAG\_COLORRESPONSEUNIT is set to 3--*Thousands*, TAG\_COLORRESPONSECURVES accepts values between 0 and 2000. If units are set to 2--*Hundredths*, TAG\_COLORRESPONSECURVES accepts values between 0 and 200.

- 1 Tenths
- 2 Hundredths
- 3 Thousandths
- 4 Ten-thousandths
- 5 Hundred-thousandths

### ***TAG\_COLORRESPONSECURVES***

ID Number (hex): 12d

Data Type: Long

This tag is used to provide further photometric interpretation information for color image data, much like gamma correction, but its use is no longer recommended.

## ***TAG\_GRAYRESPONSECURVE***

ID Number (hex): 123

Data Type: Long

TAG\_GRAYRESPONSECURVE (GRC) contains an optical density for each of the levels of gray in a grayscale image. It is analogous to Gamma curves in color images. Typical values range from an extreme white density of 0 to a very black density of 2.000 or higher. The storage of these non-integer numbers is simplified by multiplying them each by a constant and then treating them as integers. The value of TAG\_GRAYRESPONSEUNIT specifies this constant. Typically, TAG\_GRAYRESPONSEUNIT is set to 3, resulting in GRC values of 0 to 2000. If no unit is present, however, the default is 2, resulting in GRC values of 0 to 200.

The purpose of TAG\_GRAYRESPONSECURVE is to provide a photometric definition for each of the levels of gray in a grayscale image. If no GRC is present, then grayscale data should be interpreted to be linear.

## ***TAG\_GRAYRESPONSEUNIT***

ID Number (hex): 122

Data Type: Long

Because gray response curves, as defined in TAG\_GRAYRESPONSECURVE, accepts values between 0 and 2 including fractional values (e.g., 1.2 or 0.4), but the tag may only be set to integer values, TAG\_GRAYRESPONSEUNIT specifies a constants by which to multiply that number so that integer values are produced for the tag. For example, if TAG\_GRAYRESPONSEUNIT is set to 3--*Thousands*, TAG\_GRAYRESPONSECURVE accepts values between 0 and 2000. If units are set to 2--*Hundredths*, TAG\_GRAYRESPONSECURVE accepts values between 0 and 200.

- 1 Tenth
- 2 Hundredths
- 3 Thousands
- 4 Ten-thousandths
- 5 Hundred-thousandths

## ***TAG\_WHITEPOINT***

ID Number (hex): 13e

Data Type: Rational

TAG\_WHITEPOINT specifies the white point of the image to provide additional flexibility in defining full color (RGB) images. The white point is the chromaticity when each of the primaries has its Reference White value. The chromaticity value

is described using the 1931 CIE xyY chromaticity diagram, disregarding the luminance (Y) component.

This tag is a 4-byte integer, which, when considered in binary, is broken down as follows to specify the x and y values for the white point:

| Upper two bytes                              | Lower two bytes                               |
|----------------------------------------------|-----------------------------------------------|
| 0000 0000 0000 0000<br>x multiplied by 10000 | 0000 0000 0000 0000<br>y multiplied by 10 000 |

The upper two bytes specify the x component, whose value in this tag is 10 000 times the actual value. The bottom two bytes define the y component, also multiplied by 10 000. In other words, to specify the white point on the chromaticity diagram defined as (0.3127, 0.3290), TAG\_WHITEPOINT should be set to 204 934 362 (decimal), as illustrated in the table below:

| x component                          | y component                          | Tag Value                                                  |
|--------------------------------------|--------------------------------------|------------------------------------------------------------|
| 0000 1100 0011 0111<br>Decimal: 3127 | 0000 1100 1101 1010<br>Decimal: 3290 | 1100 0011 0111 0000 1100 1101 1010<br>Decimal: 204 934 362 |

## ***TAG\_PRIMARYCHROMATICITIES***

ID Number (hex): 13f

Data Type: Rational

This tag indicates the chromaticities of the primaries of the image. This is the chromaticity for each of the primaries when it has its Reference White value and the other primaries have their Reference Black values. These values are described using the 1931 CIE xy chromaticity diagram, specifying only the chromaticities not luminance. These values correspond to the filter set and light source combination of the scanner and the imaging model of the rendering algorithms. The ordering is red(x), red(y), green(x), green(y), blue(x), blue(y).

## ***TAG\_COLORMAP***

ID Number (hex): 140

Data Type: Long

This tag is used only in paletted images, and defines a Red-Green-Blue color map, or lookup table. In a paletted image, a pixel value is used to index into an RGB-lookup table. For example, a palette color pixel having a value of 0 would be displayed according to the 0<sup>th</sup> Red, Green, Blue triplet.

This tag is written, when necessary, by the scanner or ImageBASIC, and its value should not be changed by the ImageBASIC user. Doing so may dramatically alter the displayed colors of an image, usually in a detrimental fashion.

## ***TAG\_NEWSUBFILE***

ID Number (hex): 0fe

Data Type: Long

Used only in a multi-page TIFF file, this tag specifies one characteristic of the image page with which it is associated. TAG\_NEWSUBFILE indicates whether its image is a reduced resolution version of another image in the same file (1), a single page of a multi-page image (2), or a transparent mask for another image in the same file. Almost all TIFF files used in document imaging have a value of 0 for this tag, indicating that each page in the file is a separate image.

0 Separate

1 Reduced

2 Single

4 Transparency

## **Hardware and Software Descriptive Tags**

### ***TAG\_SOFTWARE***

ID Number (hex): 305

Data Type: String

This is a read-only string tag available on some scanners that will report the version and type of software that is controlling the scanner.

### ***TAG\_IMAGEDESCRIPTION***

ID Number (hex): 270

Data Type: String

This TIFF header tag may be set to a descriptive string specified by the application that is writing the TIFF file. It is currently unsupported in ImageBASIC, but may be added in a later version.

## ***TAG\_MAXTIME***

ID Number (hex): 505

Data Type: Long

TAG\_MAXTIME is a read-only tag set according to the scanner manufacturer's specifications. This tag reports the theoretical maximum time necessary to scan a single page, reported in 10<sup>ths</sup> of a second. Most scanners will actually process a page in less time than this tag indicates, so estimates of the speed and efficiency of a scanner should be based on actual tests, not the time indicated in this tag.

## ***TAG\_BUFSIZE***

ID Number (hex): 507

Data Type: Long

TAG\_BUFSIZE returns the recommended buffer size to the querying application. For ImageBASIC users, the setting of the scan buffer size is not necessary, as the application will automatically determine the optimum.

## ***TAG\_BELL***

ID Number (hex): 508

Data Type: Long

TAG\_BELL is a read-only tag that reports whether or not the scanner has a bell or other sound device. If a bell is available, a 1 is returned, otherwise a 0 is returned.

## ***TAG\_DESTINATION***

ID Number (hex): 509

Data Type: Long

This tag specifies the destination of the scanned image file. For ImageBASIC applications, this tag value should not be changed, as the destination is set with the PixScan control's properties. Altering the value of TAG\_DESTINATION may result in irregular and unpredictable activity.

- 0 Memory
- 1 File
- 2 Other

## ***TAG\_SCANNERID***

ID Number (hex): 50a

Data Type: String

TAG\_SCANNERID is a read-only ASCII tag that will return a brief string identifying the currently selected scanner. The return string is occasionally somewhat cryptic; for instance, the following values are returned for these scanners:

| Scanner               | TAG_SCANNERID   |
|-----------------------|-----------------|
| Microtek ScanMaker II | SMII            |
| Panasonic KV SS-55    | Panasonic SS 55 |
| Ricoh 520 (SCSI)      | IS520SCSI       |

## ***TAG\_PLATENLENGTH***

ID Number (hex): 50b

Data Type: Long

This read-only tag reports the actual length of the platen, which frequently corresponds to the maximum page length that the scanner is capable of processing. The value returned by querying this tag is in units of inches or TAG\_RESUNIT.

## ***TAG\_KFACTOR***

ID Number (hex): 50c

Data Type: Long

This tag indicates the type of data blocks created when compressing an image using Group 3 2-Dimensional compression (TAG\_COMPRESSION = 2 and TAG\_GROUP3OPTIONS = 1). TAG\_KFACTOR is generally set to either 2 or 4, and TAG\_ROWSPERSTRIP must be a multiple of TAG\_KFACTOR. In general, setting TAG\_COMPRESSION to Group4 will offer superior results to using Group3 compression, as well as eliminating the need to set TAG\_KFACTOR and other compression related tags.

## ***TAG\_LAMPTIMER***

ID Number (hex): 578

Data Type: Long

This tag specifies the length of time, in seconds, the scanner's lamp is lit. It may be set to an integer value from 1 to an upper value that is scanner dependent. Manipulation of this tag is discouraged as improper settings may be difficult to correct and will result in unpredictable behavior.

Default 0 use scanner's default

Exact Settings 1, 2, 3, ...

## ***TAG\_ADFTIMEOUT***

ID Number (hex): 6a5

Data Type: Long

Specifies the time in seconds the scanner will spend attempting to read the next page before reporting that no page is found. This value applies only when an ADF is used that does not support automatic page detection.

## ***TAG\_MANUALTIMEOUT***

ID Number (hex): 6a6

Data Type: Long

Specifies the time in seconds that the scanner spends attempting to read the next page before reporting that no page is found. This value only applies to scanners with manual feeders. The default value is generally between 30 and 60 seconds.

## ***TAG\_FEEDEROFFSET***

ID Number (hex): 6a7

Data Type: Long

This tag is used to make small adjustments to the currently attached feeder. Values are specified in millimeters, either positive or negative. The value in this tag moves the left edge of the scan the specified distance.

## ***TAG\_FEEDER***

ID Number (hex): 503

Data Type: Long

TAG\_FEEDER is a read-only tag that informs the application whether a page feeder (ADF) is available, and whether the scanner is able to sense a page without attempting a scan. When this tag is queried, the return value is a bitfield set according to the following values:

- 1    Feed -- The scanner has a feeder
- 2    Flat -- The scanner has a flatbed
- 10000    TellFeed -- The scanner can detect pages in the feeder
- 20000    TellFlat -- The scanner can detect a page on the flatbed
- 40000    TellScan -- A scan must be attempted to detect pages

For example, if the scanner supports a feeder and flatbed scanning, but can sense only if a page is on the flatbed but not in the ADF, a value of 20003 will be returned. Likewise, if the scanner has only a flatbed that cannot detect if a page is loaded, a value of 2 will be returned.

## ***TAG\_MAXPAGES***

ID Number (hex): 504

Data Type: Long

TAG\_MAXPAGES is a read-only tag that reports the number of pages the scanner's ADF can hold. This value is based on the physical limitations of the feeder and is only approximate. For example, if the pages being scanned are printed on heavier stock than the standard used to set this tag value, the ADF will hold fewer pages.

## ***TAG\_SCANTYPE***

ID Number (hex): 514

Data Type: Long

This tag specifies the type of scan that will be performed next. This is used most to toggle between duplex (4) and simplex (3) scanning on those scanners capable of both.

- 0 Automatic -- The default setting that means paper is fed from the feeder if there is one, otherwise the flatbed is scanned. For almost all scanners, it also scans only the front of each page.
- 1 Transparency -- For scanners with special features enabling the processing of transparencies, this setting enables those options.
- 2 Flatbed -- The single page on the flatbed will be scanned. If a **ScanBatch** is used to call the scan, the scanner will often run continuously. Use **ScanPage**, instead.
- 3 Feeder -- Front Only -- Pages are fed from the feeder and scanned simplex (front only).
- 4 Feeder -- Duplex -- Assumes a feeder is available and images are output in the order front page and then back page.
- 5 BackFront -- Assumes a feeder is present and images are output in the order back of page and then front of page.
- 6 Back Only -- Assumes a feeder is present, and only the back on each page is scanned.

## ***TAG\_SCAN Ahead***

ID Number (hex): 50d

Data Type: Byte

TAG\_SCAN Ahead may be used to specify whether a scan ahead buffer will be used in batch scanning. If scan ahead is activated, batch scanning will proceed more quickly, but consideration must be given to processing images in the scan buffer if the batch scan is stopped by the operator.

- 0 No
- 1 Yes



## ***TAG\_SCAN Ahead\_PAGES***

ID Number (hex): 515

Data Type: Long

This is a read-only tag that reports the number of pages that have been stored in the scan buffers. To specify the maximum number of pages that may be stored, set TAG\_SCAN Ahead\_MAXPAGES (&H57c).

## ***TAG\_SCAN Ahead\_MAXPAGES***

ID Number (hex): 57c

Data Type: Long

This tag sets the maximum number of pages that will be scanned ahead (before instructions from the application). If the ScanAhead property is set to False, the value of this tag is ignored. If a very large number is set, the system may run out of memory and cause an error. Typical values are around 5 pages.

## ***TAG\_CONFIGURATIONS***

ID Number (hex): 600

Data Type: String

When one of the configuration tags below is queried, TAG\_CONFIGURATIONS specifies which types of values is reported in the TAG\_CONFIGn tags. For Xionics scanners, the only values for this tag are "Default paper size:" and "Scanner Type:". Other scanners may have different values.

## ***TAG\_CONFIG\_1***

ID Number (hex): 601

Data Type: String

This ASCII tag sets the default page size. This value will be displayed in the standard **SetupScanner** dialog box, and for most scanners can be set in the **SelectScanner** dialog box through the *More...* button.

## ***TAG\_CONFIG\_2***

ID Number (hex): 602

Data Type: String

This tag is set to perform different functions depending upon the scanner model.

Xionics driven scanners will set this tag automatically, and it should not be changed by the user.

For Bell & Howell SCSI scanners, this tag specifies whether the 3-function or 5-function ACE is used. This specification can be made in the **SelectScanner** dialog through the *More...* button.

### ***TAG\_CONFIG\_n***

ID Number (hex): 60n

Data Type: String

The number of different configurations possible may range between 1 and several, and each one may have its own TAG\_CONFIGn that specifies it. In this case, the tags are sequentially numbered up to the necessary level.

### ***TAG\_FRONTPANEL***

ID Number (hex): 699

Data Type: Byte

This tag specifies whether the scanner can be controlled from the scanner's panel or only from software. When this tag is set to *Off*, many other tags (including almost all image size, contrast, and brightness tags) are reset to default values.

- 0 Off -- Software has control; front panel is disabled
- 1 On -- Front panel has control along with software

## **Image Correction and Enhancement Scanner Control Tags**

### ***TAG\_GAMMA***

ID Number (hex): 50f

Data Type: Long

Gamma correction is a means of making small color corrections in an image. It is almost always unnecessary except when processing highly detailed high color images. Gamma correction is similar to setting brightness, except that when brightness is changed, the function used to return color values maps as a straight line. When gamma is set, the function that interprets color values often maps to a curved line. The effect of this process is that, depending on a color's position within the entire spectrum, it may be written to the image file as brighter or darker than it originally seemed.

- 0 AUTO

## ***TAG\_EMPHASIS***

ID Number (hex): 510

Data Type: Short

Emphasis is a grayscale analog to gamma correction for color scanning. It individually changes the darker shades, mid-tones, and lighter shades in an image. Each of these ranges of color may be made darker or lighter separately.

- 0 Off -- Scanner output is as close to the original image as possible, based on all the other scanning parameters.
- 1 Low -- Mid-tones are made slightly brighter.
- 2 Medium -- Mid-tones are made slightly brighter, and darker shades are made slightly darker.
- 3 High -- Darker shades are made darker, mid-tones are unaffected, and lighter shades are made brighter.
- 4 Smooth -- The range of tones in the image is altered to reduce abrupt color changes between adjacent pixels.

## ***TAG\_MIXEDSCAN***

ID Number (hex): 511

Data Type: Byte

For use only for with Bell & Howell scanners with the ACE option.

- When set to *1-On*, ACE function F+1 (6), which is used for line art and photo scans, is selected.
- If any setting is made in TAG\_DITHER, then the photo mode is selected.
- The five levels of ACE sensitivity are available by setting TAG\_CONTRAST.

Valid values for this tag are as follows:

- 0 Off
- 1 On

## ***TAG\_MIRRORIMAGE***

ID Number (hex): 579

Data Type: Byte

The Fujitsu 3096G and 3097G scanners capable of outputting image files that are the mirror image of the original scanned page. If TAG\_MIRRORIMAGE is set to *1-On*, the image file that the scanner creates will be a mirror of the original. This same effect may be performed on other scanners by using TAG\_ORIENTATION.

- 0 Off
- 1 On

## ***TAG\_MAXBUFSIZE***

ID Number (hex): 581

Data Type: Long

For some Microtek scanners, this tag reports the maximum size buffer in bytes that the scanner is able to use. Regardless of the value in this tag, however, it has never become necessary here to change any buffer size to accommodate any particular scanner.

## ***TAG\_DROPOUT***

ID Number (hex): 582

Data Type: Short

For those scanners capable of filtering out printer's inks in dropout colors, this tag must be set to indicate which color to omit in the scan output. If this tag is set to *None* and dropout inks are used, the scanner may or may not detect them, depending upon its sensitivity and the quality of the printing.

- 0 None
- 1 Red
- 2 Green
- 3 Blue

Most scanners require optional light sources (lamps) to drop out some colors. Your scanner's documentation should indicate whether your specific scanner operates in this manner, as well as the necessary lamp color to drop out a given printer's ink.

## ***TAG\_OUTLINE***

ID Number (hex): 5a6

Data Type: Byte

This tag controls the activation of outline extraction. If this option is activated, TAG\_EMPHASIS (and TAG\_MIXEDSCAN on Bell & Howell) is disabled.

- 0 Off
- 1 On

## ***TAG\_DETECTJOBSEP***

ID Number (hex): 5a7

Data Type: Short

For those scanners with a job detection option, this tag is set to enable or disable the process, as well as instruct the scanner how to proceed when a separator page is found.

- 0 Off -- No separator detection
- 1 NoStop\_NoEject -- Activate detection and include the separator page in the scan.
- 2 NoStop\_Eject -- Activate detection and eject the separator page without scanning it.
- 3 Stop\_NoEject -- Activate detection and stop scanning without ejecting the separator.
- 4 Stop\_Eject -- Activate detection and stop scanning after ejecting the separator page.

## ***TAG\_JOBSEPDETECTED***

ID Number (hex): 5a8

Data Type: Long

If TAG\_DETECTJOBSEP is activated, and a separator page is detected during scanning, TAG\_JOBSEPDETECTED will be set to a non-zero value. This tag is set simply to inform the developer that a separator page has been found, allowing the inclusion of code to perform any special actions that may be necessary. For example, either a new multi-page file should be started, or the operator may wish to be informed of the detection so that the scanned pages may be removed from the scanner and filed together.

## ***TAG\_DTCSELECTION***

ID Number (hex): 5a0

Data Type: Short

This tag controls the optional dynamic threshold control (DTC) function. This is a new feature to the 3096G. Legal values include:

- 0 Disable DTC; use TAG\_BRIGHTNESS to set threshold
- 1 Use simplified DTC; use TAG\_DTCVARIANCE to set variance rate
- 2 Use standard DTC; use other tags described below for IPC II only

Disabling DTC means normal thresholding is used; this is controlled through TAG\_BRIGHTNESS. Enabling DTC will over-ride any TAG\_BRIGHTNESS setting.

- If TAG\_DTCSELECTION is 0, TAG\_BRIGHTNESS is operative
- If TAG\_DTCSELECTION is 1, TAG\_DTCVARIANCE is operative
- If TAG\_DTCSELECTION is 2, TAG\_THRESHOLDTRACKING, TAG\_DTCGRADATION, TAG\_DTCSMOOTHING, TAG\_DTCFILTER, TAG\_PIXELPATCH are operative.

### **TAG\_THRESHOLDTRACKING**

ID Number (hex): 586

Data Type: Short

For the Fujitsu IPC II board, this tag controls the threshold level when TAG\_DTCSELECTION is set to *Standard DTC*. In this case, the valid options are 1 through 7. Other scanners may support other options.

- |   |       |                                                          |
|---|-------|----------------------------------------------------------|
| 1 | 37    | light (37%) tracking point                               |
| 2 | 75    | dark (75%) tracking point                                |
| 3 | 75_75 | dark (75%) setting, tracking upper limit of 75%          |
| 4 | 75_85 | dark (75%) setting, tracking upper limit of 85%          |
| 5 | 75_90 | dark (75%) setting, tracking upper limit of 90%          |
| 6 | 62_80 | slightly dark (62%) setting, tracking upper limit of 80% |
| 7 | 62_90 | slightly dark (62%) setting, tracking upper limit of 90% |

### **TAG\_DTCVARIANCE**

ID Number (hex): 5a1

Data Type: Short

For simple DTC, this tag specifies the acceptable variation. For the Fujitsu 3097G with IPC II, the range is from 0 to 7, inclusive.

### **TAG\_DTCGRADATION**

ID Number (hex): 5a2

Data Type: Short

This tag specifies how smooth the gradation of image data will be across relatively near-by pixels when using the IPC II. The *0--Normal* setting generally provides acceptable output, but some images may require more extensive manipulation during processing. *1--High* allows this more extended processing.

- |   |                     |
|---|---------------------|
| 0 | Normal image        |
| 1 | High contrast image |

### ***TAG\_DTCSMOOTHING***

ID Number (hex): 5a3

Data Type: Short

This tag controls the smoothing mode parameter for standard DTC on the IPC II.  
Legal values are as follows:

- 0 OCR
- 1 "Image Scanner"

### ***TAG\_DTCFILTER***

ID Number (hex): 5a4

Data Type: Short

This tag controls the Filtering parameter for standard DTC on the IPC II. Legal values are as follows:

- 0 Ball point pen mode
- 1 Normal mode

### ***TAG\_PIXELPATCH***

ID Number (hex): 589

Data Type: Long

This tag controls "Noise Removing" for standard DTC on the IPC II. Legal values for this tag range from 0 to 3F(hex); it should be set as six bit fields. These bit fields are:

- 1 On indicates threshold-value for binary output is black;  
Off indicates output is white
- 2 Noise removing of 2x2 matrix
- 4 Noise removing of 3x3 matrix
- 8 Noise removing of 4x4 matrix
- 16(dec) Noise removing of 5x5 matrix
- 32(dec) On => noise removing disabled  
Off => noise removing enabled

## Endorser Support Scanner Control Tags

### ***TAG\_ENDORSER\_INCSTART***

ID Number (hex): 584

Data Type: Long

This integer tag specifies the starting point for auto-incrementing during a batch scan. Starting at this point, each page scanned will increment the count by one. This tag is used by both the InkJet and Electronic endorsers of the Ricoh 5x0 scanners, and may be available on other models.

### ***TAG\_ENDORSER\_STRING***

ID Number (hex): 583

Data Type: String

This ASCII tag holds the string that is to be printed on each page as it is scanned. Some scanners endorse scanned pages by printing on the page itself as it is scanned. Others embed the endorser string in the output image file. A few, such as the Ricoh 520 offer both options. The type of endorsement available and limits on the string size and characters should be addressed in the documentation shipped with your scanner.

Ricoh 510/520 -- For InkJet endorsing, the string may be up to 80 characters long and is printed as a single column running with the scan direction. This print format is created because the movement of the paper is used rather than moving the printing head.

The first occurrence of a sequence of the special character "%" will be replaced with the page count (cf. TAG\_ENDORSER\_INCSTART). For example, if this tag is set as shown in the next line,

```
PixScan1.SetTagString &H583, "REFERENCE ID# DHS_%%%"
```

the string on the scanned page would be printed as follows for page count 126:

```
REFERENCE ID# DHS_00126
```

The admissible characters are <space> - \_ ^ # . ] [ ( ) = + < > Aa-Zz 0-9.



## ***TAG\_ENDORSER\_YOFFSET***

ID Number (hex): 585

Data Type: Long

This tag specifies the distance from the top edge of the page to the beginning of the endorser string. Changing the horizontal location of the endorser string is accomplished by manually moving the endorser cartridge to the appropriate position.

## ***TAG\_SPASPEED\_200***

ID Number (hex): 591

Data Type: Long

This tag and the following TAG\_SPA... tags specify the EEPROM settings for Ricoh 510/520 InkJet endorser (SPA). These settings change the speed at which the endorser prints, allowing for different speeds depending upon the scanning resolution. Changing these tag values will cause the endorser to print more quickly or more slowly, stretching or compressing the final length of the string. Care must be taken when changing these values because small change -- as little as 20 to 25 -- can make the endorsed string illegible.

Before changing these values, record the current settings so you may return the scanner to its default settings if necessary. These are the speed and base offsets at various resolutions. Note that the driver requires that 200-400 be consecutive. At the time of this writing, the default value for TAG\_SPASPEED\_200 is 213.

## ***TAG\_SPASPEED\_300***

ID Number (hex): 592

Data Type: Long

TAG\_SPASPEED\_300 specifies the speed at which the InkJet endorser will print on the scanned page when scanning at 300 DPI. The default value at the time of this writing is 190. Higher integers will cause the endorser to write more slowly, stretching the printed output, while lower numbers will compress the length of the printed string.

## ***TAG\_SPASPEED\_400***

ID Number (hex): 593

Data Type: Long

TAG\_SPASPEED\_400 specifies the speed at which the InkJet endorser will print on the scanned page when scanning at 400 DPI. The default value at the time of this writing is 172. Higher integers will cause the endorser to write more slowly, stretching the printed output, while lower numbers will compress the length of the printed string.

### ***TAG\_SPAOFF\_200***

ID Number (hex): 594

Data Type: Long

TAG\_SPAOFF\_200 specifies the distance in pixels from the top of the page to the beginning of the endorser string. This tag specifies the vertical offset when printing at 200 DPI, and the following two tags specify this offset at other scanning resolutions.

### ***TAG\_SPAOFF\_300***

ID Number (hex): 595

Data Type: Long

TAG\_SPAOFF\_300 specifies the distance in pixels from the top of the page to the beginning of the endorser string when scanning at 300 DPI. This tag specifies the vertical offset when printing at 300 DPI, and other tags specify this offset at other scanning resolutions.

### ***TAG\_SPAOFF\_400***

ID Number (hex): 596

Data Type: Long

TAG\_SPAOFF\_400 specifies the distance in pixels from the top of the page to the beginning of the endorser string when scanning at 400 DPI. This tag specifies the vertical offset when printing at 400 DPI, and other tags specify this offset at other scanning resolutions.

### ***TAG\_ELECENDORSER\_STRING***

ID Number (hex): 700

Data Type: String

For Electronic Endorsing, up to eight lines of 256 characters may be included. However, the actual number of lines and characters that can be printed may be limited by string position, font size, and marking zoom factor.

The first occurrence of a sequence of the special character "%" will be replaced with the page count (cf. TAG\_ENDORSER\_INCSTART). For example, the string

REFERENCE ID# DHS\_%%%

would be printed as the following string for page count 126:

REFERENCE ID# DHS\_00126

The admissible characters are <space> - \_ ^ # . ] [ ( ) = + < > Aa-Zz 0-9.

### ***TAG\_ELECENDORSER\_YOFFSET***

ID Number (hex): 701

Data Type: Long

This tag specifies the vertical distance in pixels from the top of the image to the beginning of the endorser string. If insufficient space is left to add the entire endorser string to the image, the excess will be lost.

### ***TAG\_ELECENDORSER\_ZOOM***

ID Number (hex): 702

Data Type: Short

The default size of the text added by the electronic endorser is approximately 10 point at 300 DPI. This size may be altered by setting TAG\_ELECENDORSER\_ZOOM to increase the text size. The only valid settings for this tag are 1, 2, 4, and 8, each of which is the multiple by which the text size is increased. For example, setting this tag equal to 4 would produce 40 point text at 300 DPI.

### ***TAG\_ELECENDORSER\_MODE***

ID Number (hex): 703

Data Type: Short

This tag specifies the color of the electronic endorser string bitmap as it is added to the image file. If black stamp or white stamp is selected, all the text added will be of that color. If preserve is selected, the endorser text will be white if it is placed over a black image region, and black if it is placed over a white image region.

- 0 Preserve
- 1 Black Stamp
- 2 White Stamp

## Image Quality Options through Scanner Control Tags

### ***TAG\_THRESHOLDTRACKING***

ID Number (hex): 586

Data Type: Long

For the Fujitsu IPC II board, this tag controls the threshold level when TAG\_DTCSELECTION is set to 2--*Standard DTC*. In this case, the valid options are 1 through 7. Other scanners may support other options.

- 0 Midpoint
- 1 light (37%) tracking point
- 2 dark (75%) tracking point
- 3 dark (75%) setting, tracking upper limit of 75%
- 4 dark (75%) setting, tracking upper limit of 85%
- 5 dark (75%) setting, tracking upper limit of 90%
- 6 slightly dark (62%) setting, tracking upper limit 80%
- 7 slightly dark (62%) setting, tracking upper limit 90%
- 8 TBD1
- 9 TBD2
- 10 TBD3
- 11 TBD4
- 12 dark (75%) setting, tracking upper limit of 85%
- 13 light (37%) setting, tracking upper limit of 85%
- 14 slightly dark (62%) setting tracking upper limit 80%
- 15 light (31%) setting, tracking upper limit of 80%

### ***TAG\_MATRIX***

ID Number (hex): 587

Data Type: Short

- 0 Combines threshold, high and low frequency enhancement
- 1 Output value of JP4.2
- 2 Disable high frequency enhancement
- 3 PASSTHROUGH
- 4 Combines threshold, low and more high freq. enhancement
- 5 JP42\_2
- 6 HIGHOFF\_2
- 7 PASSTHROUGH\_2

## ***TAG\_ENHANCE***

ID Number (hex): 588

Data Type: Short

- 0 LESS -- less sensitive to changes
- 1 SPECIAL1 -- special case 1
- 2 SPECIAL2 -- special case 2
- 3 MORE -- more sensitive to changes
- 4 NONOISEBLOCK -- noise block off
- 5 NOISEBLOCK -- noise block on

## **EEPROM Settings through Scanner Control Tags**

All the offset tags in this section specify the vertical offset from the leading edge of the page to the start of the actual scan. Each of these tags specifies an offset at a single resolution, allowing preset values to be maintained when scanning resolution is changed. Different front and back offsets are available for each resolution, allowing the selection of different scan areas on the front and back of each page. These are default settings and may be changed when scanning is actually begun by setting TAG\_YPOSITION. Horizontal offsets may be specified similarly by setting TAG\_XPOSITION. Note that the driver requires that 200-400 be consecutive.

### ***TAG\_FRONTOFF\_200***

ID Number (hex): 58a

Data Type: Long

This tag specifies the offset from the leading edge of the scanned page to the start of the scanning process when scanning the front of a page at 200 DPI.

### ***TAG\_FRONTOFF\_300***

ID Number (hex): 58b

Data Type: Long

This tag specifies the offset in pixels from the leading edge of the scanned page to the start of the scanning process when scanning the front of a page at 300 DPI.

### ***TAG\_FRONTOFF\_400***

ID Number (hex): 58c

Data Type: Long

This tag specifies the offset in pixels from the leading edge of the scanned page to the start of the scanning process when scanning the front of a page at 400 DPI.

### ***TAG\_BACKOFF\_200***

ID Number (hex): 58d

Data Type: Long

This tag specifies the offset in pixels from the leading edge of the scanned page to the start of the scanning process when scanning the back of a page at 200 DPI.

### ***TAG\_BACKOFF\_300***

ID Number (hex): 58e

Data Type: Long

This tag specifies the offset in pixels from the leading edge of the scanned page to the start of the scanning process when scanning the back of a page at 300 DPI.

### ***TAG\_BACKOFF\_400***

ID Number (hex): 58f

Data Type: Long

This tag specifies the offset in pixels from the leading edge of the scanned page to the start of the scanning process when scanning the back of a page at 400 DPI.

### ***TAG\_XBASEOFFSET***

ID Number (hex): 590

Data Type: Long

This tag may be set to specify a standard horizontal offset from the left edge of the page being scanned. This value is in pixels, and is therefore resolution dependent, so care should be taken to ensure a correct value is set in this tag before scanning is begun.

## **Window and Subwindow Access Tags**

### ***TAG\_WINDOW***

ID Number (hex): 69a

Data Type: Long

This tag is used to select a bank of different tag values required to support scanners that are capable of independently definable sections, regions, or sub-windows. Whenever TAG\_WINDOW is available, its range of valid values indicates the number of separate regions that the scanner is capable of supporting. The image contained in each of these regions may be accessed independently after scanning a page by setting TAG\_WINDOW to the value of the region of interest and then reading the image as usual. If a scanner uses TAG\_SUBWINDOW instead of

TAG\_WINDOW, separate regions will not be available for reading, but each region in the main scan can have different parameters to allow a highly configurable mixed scanning mode.

Both TAG\_WINDOW and TAG\_SUBWINDOW must be used within strict guidelines that may vary from scanner to scanner, but the following guidelines apply to most scanner models.

- The tags shown in the table below may each be given a different value in each window.
- To set these values, first select the window by setting TAG\_WINDOW, then set each tag value that will be different in that window.
- It is advisable to query all these tags as each window is being defined to ensure that it is set to the proper value.
- Matching windows on the front and back of a page (for instance windows 2 and -2) may differ by a limited amount. For example, few scanners support radically different resolutions in these windows -- one window may have a resolution one-half or one-third the other, but that is usually the limit.

The following table lists the tags with multiple values and the Windows where they are tracked. Although the table shows window ID numbers from -9 to 9, not all scanners support this many different windows. Some scanners allow only windows -1 and 1, which identify the back and front of the page, respectively. Other scanners will support between 2 and 9 windows for each side of the page, with the back windows usually given negative numbers.

| Tag               | Window -1 to 1 (or 0 to 1) |
|-------------------|----------------------------|
| TAG_XPOSITION     | linked to 1                |
| TAG_YPOSITION     | linked to 1                |
| TAG_IMAGEWIDTH    | linked to 1                |
| TAG_IMAGELENGTH   | linked to 1                |
| TAG_BRIGHTNESS    | yes                        |
| TAG_CONTRAST      | yes                        |
| TAG_DITHER        | yes                        |
| TAG_MIXEDSCAN     | yes                        |
| TAG_COMPRESSION   | linked to 1                |
| TAG_GROUP3OPTIONS | yes                        |
| TAG_KFACTOR       | linked to 1                |

| Tag | Window -9 to -2 | Window 2 to 9 |
|-----|-----------------|---------------|
|-----|-----------------|---------------|

|                   |                                                   |     |
|-------------------|---------------------------------------------------|-----|
| TAG_XPOSITION     | linked to front window of same number             | yes |
| TAG_YPOSITION     | linked to front window of same number             | yes |
| TAG_IMAGEWIDTH    | linked to front window of same number             | yes |
| TAG_IMAGELENGTH   | linked to front window of same number             | yes |
| TAG_BRIGHTNESS    | no                                                | no  |
| TAG_CONTRAST      | no                                                | no  |
| TAG_DITHER        | no                                                | no  |
| TAG_MIXEDSCAN     | no                                                | no  |
| TAG_COMPRESSION   | linked to front window of same number             | yes |
| TAG_GROUP3OPTIONS | linked to front window of same number linked to 1 |     |
| TAG_KFACTOR       | linked to front window of same number             | yes |

### ***TAG\_WINDOW\_COUNT\_BACK***

ID Number (hex): 6ab

Data Type: Long

Reports the number of windows defined for the back page. Applies to scanners that support both duplex scanning and window definition.

### ***TAG\_WINDOW\_COUNT\_FRONT***

ID Number (hex): 6ac

Data Type: Long

This tag reports the number of windows defined for the front page. This only applies to scanners that support both duplex scanning and window definition.

### ***TAG\_SUBWINDOW***

ID Number (hex): 6ad

Data Type: Long

Those scanners that support multiple region definition for a scan will support either TAG\_WINDOW or TAG\_SUBWINDOW. These tags are set and operate similarly, with the exception that the scanners supporting TAG\_SUBWINDOW output a single image containing all the regions, forcing the user to simply view the



image as a whole, unlike TAG\_WINDOW support in which each window may be viewed separately. Except for this difference, TAG\_SUBWINDOW is set just like TAG\_WINDOW, above, with the same method of numbering regions for the front with positive numbers and regions on the back of the page with negative numbers. The same tags shown in the TAG\_WINDOW table are also tracked with TAG\_SUBWINDOW.

### ***TAG\_SUBWINDOW\_COUNT\_BACK***

ID Number (hex): 6ae  
Data Type: Long

This tag reports the number of subwindows defined for the back page. This only applies to scanners that support both duplex scanning and subwindow definition.

### ***TAG\_SUBWINDOW\_COUNT\_FRONT***

ID Number (hex): 6af  
Data Type: Long

This tag reports the number of subwindows defined for the front page. This only applies to scanners that support both duplex scanning and subwindow definition.

## **Bar Code Recognition**

Bar code recognition through the Bell & Howell RSC, some Kofax boards, and other hardware devices is controlled through the following tags. The information reported through these tags is not available on all hardware. Enabling and reading bar code recognition results is performed the same on all hardware, as shown below:

```
' enable hardware assisted bar code recognition
' through TAG_BARCODE
PixScan1.SetTagLong(&H69b, 0, 1)

' enable reading of front of page through
' TAG_BAR_AUTOREADFRONT
PixScan1.SetTagLong(&H6c0, 0, 1)

' enable reading of front of page through
' TAG_BAR_AUTOREADBACK
PixScan1.SetTagLong(&H6c1, 0, 1)

' perform a scanning operation
PixScan1.ScanPage

' read recognition results from TAG_BARDATA_TEXT
bc_text = PixScan1.GetTagString(&H6c7, 0)
```

**Note:** All of the bar code related tags whose names begin with `TAG_BAR_...` are used to set parameters to the bar code recognition. All of the bar code related tags whose names begin with `TAG_BARDATA_...` report the recognition results.

## ***TAG\_BARCODE***

ID Number (hex): 69b

Data Type: Byte

`TAG_BARCODE` specifies whether or not the scanner's bar code recognition hardware is active. If this tag is set to `0--Off`, the other bar data control tags will be ignored. If bar code recognition is enabled, several other tags must be set to instruct the recognition hardware on which type of bar code to find and various limits on the search parameters. When one or more bar codes are found, several other tags are filled with information about the bar code.

0 Off

1 On

The search parameter tags that must be set prior to scanning:

`TAG_BAR_AUTOREADBACK`

`TAG_BAR_AUTOREADFRONT`

`TAG_BAR_MAXCOUNT`

`TAG_BAR_MAXRATIO`

`TAG_BAR_MINCOUNT`

`TAG_BAR_MINHEIGHT`

`TAG_BAR_SEARCHCOUNT`

`TAG_BAR_SEARCHMODE`

`TAG_BAR_SEARCHTIME`

`TAG_BAR_TYPE`

`TAG_BAR_TYPECOUNT`

The search result tags that report the details of each bar code that is found:

`TAG_BARDATA_LENGTH`

`TAG_BARDATA_ORIENTATION`

`TAG_BARDATA_SEARCHTIME`

`TAG_BARDATA_TEXT`

`TAG_BARDATA_TYPE`

`TAG_BARDATA_WIDTH`

`TAG_BARDATA_XPOSITION`

`TAG_BARDATA_YPOSITION`

## ***TAG\_BAR\_TYPE***

ID Number (hex): 69c

Data Type: Long

TAG\_BAR\_TYPE specifies the type of bar code that the scanner will attempt to locate and recognize. If a different type of bar code is present, it may or may not be located, but will certainly be read improperly if it is found. The following list of bar code type shows all the currently supported formats.

- 0 No bar code
- 1 Type EAN-8
- 2 Type EAN-13
- 3 Type CODE 39
- 4 CODE 2 of 5, interleaved
- 5 CODE 2 of 5, 3 lines (matrix)
- 6 CODE 2 of 5, 3 lines (datalogic)
- 7 CODE 2 of 5, 5 lines (industrial)
- 8 Patch Code type II, III, or T

## ***TAG\_BAR\_TYPECOUNT***

ID Number (hex): 69d

Data Type: Long

This is a read-only tag that reports the maximum number of bar codes that may be searched for during scanning. The value of this tag should be queried before setting TAG\_BAR\_TYPE and used to limit the number of options that may be set in that tag.

## ***TAG\_BAR\_MINHEIGHT***

ID Number (hex): 69e

Data Type: Long

This tag sets a minimum height that must be reached by a bar code before the scanner will attempt to recognize it. Any bar code that is not at least this tall will be ignored, so very small values in this tag will increase processing time by forcing the scanner to attempt to recognize all small rectangular regions that may be bar codes. This tag is set in millimeters and defaults to 5.

## ***TAG\_BAR\_SEARCHCOUNT***

ID Number (hex): 69f

Data Type: Long

This tag specifies the number of attempts the decoding algorithm will make before failing to recognize a bar code. For example, if TAG\_BAR\_SEARCHCOUNT is set to 5, the scanner will try five times to recognize each bar code it has located. If it cannot recognize the bar code in this number of cycles, it will simply fail.

TAG\_BAR\_SEARCHCOUNT must be set to an integer value between 0 and 7, inclusive, with 0 indicating the scanner's default. Lower values in this tag will speed processing, but may result in inaccurate results if scanning lower quality bar codes.

### ***TAG\_BAR\_SEARCHMODE***

ID Number (hex): 6a0

Data Type: Long

This tag specifies the search direction during decoding, and the scanner will recognize bar codes that are rotated less than 45 degrees from the search direction. For example, if it is known that all bar codes are read horizontally, then this tag may be set to *0--Horizontal*. If bar codes will be presented in a variety of orientations, the search mode should be set to both horizontal and vertical to allow recognition of all.

- 0 Horizontal
- 1 Vertical
- 2 Vertical then Horizontal
- 3 Horizontal then Vertical

### ***TAG\_BAR\_SEARCHTIME***

ID Number (hex): 6a1

Data Type: Long

This tag specifies the maximum time that may be spent searching for bar codes. If this time limit is reached without finding a bar code, control is returned to the application. This tag may be set to any integer between 20 and 65536 milliseconds. 0 may also be set to disable the time-out feature.

### ***TAG\_BAR\_MAXRATIO***

ID Number (hex): 6a2

Data Type: Long

When multiple bar codes are being recognized, this tag specifies the maximum variation between the widest and shortest bar codes. Values are generally between 3 and 6, and default values vary depending upon the type of bar code. Setting TAG\_BAR\_TYPE overwrites this value with the default for the newly specified type.

### ***TAG\_BAR\_MINCOUNT***

ID Number (hex): 6a3

Data Type: Long

This tag specifies the minimum number of bars in a bar code, and any bar code not matching this minimum will not be recognized. The value of this tag is reset to a default (which varies according to type) when TAG\_BAR\_TYPE is changed.

### ***TAG\_BAR\_MAXCOUNT***

ID Number (hex): 6a4

Data Type: Long

This tag specifies the maximum number of bars in a bar code, and any bar code exceeding this limit will not be recognized. The value of this tag is reset to a default (which varies according to type) when TAG\_BAR\_TYPE is changed.

### ***TAG\_BAR\_AUTOREADFRONT***

ID Number (hex): 6c0

Data Type: Byte

If this tag is set to 1 when scanning begins, the front of the page is automatically searched for a bar code based on the parameters set in the tags shown above. To minimize search time, a window that defines a relatively small region around the bar code should be selected.

0 Off

1 On

## ***TAG\_BAR\_AUTOREADBACK***

ID Number (hex): 6c1

Data Type: Byte

If this tag is set to 1 when scanning begins, the back of the page is automatically searched for a bar code based on the parameters set in the tags shown above. To minimize search time, a window that defines a relatively small region around the bar code should be selected.

0 Off

1 On

## ***TAG\_BARDATA\_TYPE***

ID Number (hex): 6c2

Data Type: Short

This tag reports the type of bar code that was recognized. An integer value is returned when this tag is queried, according to the list below:

- 1 Type EAN-8
- 2 Type EAN-13
- 3 Type CODE 39
- 4 CODE 2 of 5, interleaved
- 5 CODE 2 of 5, 3 lines (matrix)
- 6 CODE 2 of 5, 3 lines (datalogic)
- 7 CODE 2 of 5, 5 lines (industrial)
- 8 Patch Code type II, III, or T

## ***TAG\_BARDATA\_XPOSITION***

ID Number (hex): 6c3

Data Type: Long

This tag reports the horizontal position of the left edge of the bar code. Because this tag reports position in pixels, it is resolution dependent, so any processing of this information should also take into account the resolution of the scan.

## ***TAG\_BARDATA\_YPOSITION***

ID Number (hex): 6c4

Data Type: Long

This tag reports the vertical position of the top edge bar code. Because this tag reports position in pixels, it is resolution dependent, so any processing of this information should also take into account the resolution of the scan.

### ***TAG\_BARDATA\_WIDTH***

ID Number (hex): 6c5

Data Type: Long

This tag reports the width of the recognized bar code. The width is a measure of the distance perpendicular to the length of the bar code:



### ***TAG\_BARDATA\_LENGTH***

ID Number (hex): 6c6

Data Type: Long

This tag reports the length in pixels of the recognized bar code.

### ***TAG\_BARDATA\_TEXT***

ID Number (hex): 6c7

Data Type: String

After any bar code is recognized, TAG\_BARDATA\_TEXT is populated with the resulting string. This tag should be queried and processed immediately after recognition to avoid losing this information:

```
sBarValue = PixScan1.GetTagString(&H6c7, 0)
```

### ***TAG\_BARDATA\_ORIENTATION***

ID Number (hex): 6c8

Data Type: Long

This tag reports the orientation of the bar code relative to the page. Most scanners can read bar codes in any orientation, so this tag is supplied for information purposes only.

- 1 Portrait -- Page and bar code are oriented the same
- 2 Landscape -- Bar code reads from the top of the page down
- 3 180° -- Bar code is upside-down
- 4 270° -- Bar code is sideways and reads from the bottom of the page up

### ***TAG\_BARDATA\_SEARCHTIME***

ID Number (hex): 6c9

Data Type: Long

Reports the number of milliseconds that the scanner or RSC required to locate and process this bar code.



# Index

## A

- AboutBox Method 21
- Active Property 21
- Appending Output File 17
- Appending Output Files 31
- Application Design Guidelines 5
- ASPI Drivers 7

## B

- Bar Code Support
  - TAG\_BAR\_AUTOREADBACK 98
  - TAG\_BAR\_AUTOREADFRONT 97
  - TAG\_BAR\_MAXCOUNT 97
  - TAG\_BAR\_MAXRATIO 97
  - TAG\_BAR\_MINCOUNT 97
  - TAG\_BAR\_MINHEIGHT 95
  - TAG\_BAR\_SEARCHCOUNT 96
  - TAG\_BAR\_SEARCHMODE 96
  - TAG\_BAR\_SEARCHTIME 96
  - TAG\_BAR\_TYPE 95
  - TAG\_BAR\_TYPECOUNT 95
  - TAG\_BARCODE 94
  - TAG\_BARDATA\_LENGTH 99
  - TAG\_BARDATA\_ORIENTATION 99
  - TAG\_BARDATA\_SEARCHTIME 100
  - TAG\_BARDATA\_TEXT 99
  - TAG\_BARDATA\_TYPE 98
  - TAG\_BARDATA\_WIDTH 99
  - TAG\_BARDATA\_XPOSITION 98
  - TAG\_BARDATA\_YPOSITION 98
- Batch Processing Events 20, 41
- Batch Scanning 18
- Batch Scanning, Initiating 35
- BMP File Format 43
- Buffer Size
  - TAG\_MAXBUFSIZE 80
- Buffers, clearing 19, 22
- Buffers, Scan Ahead 34

## C

- CALS File Format 44
- Canceling a Batch 18, 20
- CancelScan Parameter 18, 20

- Clear Method 22
- ClearScanAhead Method 19, 22
- Color Definition
  - TAG\_BITSPERSAMPLE 63
  - TAG\_BLUEBRIGHTNESS 62
  - TAG\_GREENBRIGHTNESS 62
  - TAG\_GREENCONTRAST 61
  - TAG\_PHOTOMETRICINTERPRETATION 65
  - TAG\_REDBRIGHTNESS 62
  - TAG\_REDCONTRAST 61
  - TAG\_SAMPLESPERPIXEL 64
- Compressed Data from Scanner
  - UseScannerCompression Property 13, 40
- Compression
  - TAG\_KFACTOR 74
- Configuration
  - TAG\_CONFIG\_1 77
  - TAG\_CONFIG\_2 77
  - TAG\_CONFIG\_n 78
  - TAG\_CONFIGURATIONS 77
- Configuration, Loading
  - RestoreTagSettings Method 13, 33
- Configuration, Saving
  - SaveTagSettings Method 13, 34
- ConfigureScanner Method 23
- Configuring the Scanner
  - SetupScanner Method 11
  - Tag Interface 12
- Control Arrays
  - DynamicControl Property 23
- Count of Pages 20

## D

- Data Format
  - TAG\_COMPRESSION 65
  - TAG\_FILLORDER 67
  - TAG\_GROUP3OPTIONS 66
  - TAG\_GROUP4OPTIONS 67
  - TAG\_PLANARCONFIGURATION 67
  - TAG\_ROWSPERSTRIP 68
  - TAG\_STRIPBYTECOUNTS 68
  - TAG\_STRIPOFFSETS 68
- Data Input
  - ImageDataSource Property 1
- Data Interpretation
  - TAG\_COLORMAP 71
  - TAG\_COLORRESPONSECURVES 69
  - TAG\_COLORRESPONSEUNIT 69
  - TAG\_GAMMA 78

- TAG\_GRAYRESPONSECURVE 70
- TAG\_GRAYRESPONSEUNIT 70
- TAG\_NEWSUBFILE 72
- TAG\_ORIENTATION 68
- TAG\_PRIMARYCHROMATICITIES 71
- TAG\_SCANORIENTATION 60
- TAG\_WHITEPOINT 70

- Data Types, GetTagType Method 12

- Defaults, setting

- SetTagDefault Method 12

- Development Licensing 4

- Dialog

- Scanner Select 8

- Scanner Setup 11

- DidScan Event 20, 23

- Driver Name 35

- Scanner Property 9, 10

- Drivers, System-level 7

- Duplex Scanning

- TAG\_SCANTYPE 76

- Dynamic Creation of Controls

- DynamicControl Property 23

- DynamicControl Property 23

## E

- EEPROM Settings

- TAG\_BACKOFF\_200 90

- TAG\_BACKOFF\_300 90

- TAG\_BACKOFF\_400 90

- TAG\_FRONTOFF\_200 89

- TAG\_FRONTOFF\_300 89

- TAG\_FRONTOFF\_400 89

- TAG\_XBASEOFFSET 90

- Elements in Tag, GetTagLength Method 12

- Endorser

- TAG\_ELECENDORSER\_MODE 87

- TAG\_ELECENDORSER\_STRING 86

- TAG\_ELECENDORSER\_YOFFSET 87

- TAG\_ELECENDORSER\_ZOOM 87

- TAG\_ENDORSER\_INCSTART 84

- TAG\_ENDORSER\_STRING 84

- TAG\_ENDORSER\_YOFFSET 85

- TAG\_SPAOFF\_200 86

- TAG\_SPAOFF\_300 86

- TAG\_SPAOFF\_400 86

- TAG\_SPASPEED\_200 85

- TAG\_SPASPEED\_300 85

- TAG\_SPASPEED\_400 85

- Enhancement

- TAG\_BRIGHTNESS 61

- TAG\_CONTRAST 61

- TAG\_DITHER 65

- Error Event 23

- Event Parameters

- CancelScan 18

- Events

- DidScan 20, 23

- Error 23

- ImageDataChanged 2

- WillScan 18, 20, 41

## F

- Feeder

- TAG\_ADFTIMEOUT 75

- TAG\_FEEDER 75

- TAG\_FEEDEROFFSET 75

- TAG\_MAXPAGES 76

- File Name of Output 32

- File Writing

- OutputFileAppend 16

- OutputFileFormat Property 16

- OutputFileName Property 16

- Format of Output Files 31

- Front Panel Control

- TAG\_FRONTPANEL 78

## G

- GetSavedScanner Method 9, 10, 24

- GetTagChoiceCount Method 24

- GetTagChoiceFlags Method 24

- GetTagChoiceLong Method 25

- GetTagChoiceRational Method 26

- GetTagChoiceString Method 26

- GetTagLength Method 12, 27

- GetTagLengthDefault Method 27

- GetTagLong Method 12, 27

- GetTagLongDefault Method 12, 28

- GetTagRational Method 12, 28

- GetTagRationalDefault 12

- GetTagRationalDefault Method 12, 28

- GetTagString Method 13, 29

- GetTagStringDefault Method 29

- GetTagType Method 12, 29, 51

- GIF File Format 44

- Group 3 Compression 45

- Group 4 Compression 45

## H

Hardware Setup 7

## I

Image Enhancement

- TAG\_DROPOUT 80
- TAG\_DTCFILTER 83
- TAG\_DTCGRADATION 82
- TAG\_DTCSELECTION 81
- TAG\_DTCSMOOTHING 83
- TAG\_DTCVARIANCE 82
- TAG\_EMPHASIS 79
- TAG\_ENHANCE 89
- TAG\_GAMMA 63
- TAG\_MATRIX 88
- TAG\_MIRRORIMAGE 79
- TAG\_MIXEDSCAN 79
- TAG\_OUTLINE 80
- TAG\_PIXELPATCH 83
- TAG\_THRESHOLDTRACKING 82, 88

ImageDataChanged Event 2

ImageDataSource Property 1, 15

Informational Tags

- TAG\_BELL 73
- TAG\_BUFSIZE 73
- TAG\_DESTINATION 73
- TAG\_IMAGEDESCRIPTION 72
- TAG\_LAMPTIMER 74
- TAG\_MAXTIME 73
- TAG\_PLATENLENGTH 74
- TAG\_SCANNERID 74
- TAG\_SOFTWARE 72

Initiating Scanning

ScanPage Method 17

Installing a Scanner 7

IsPageLoaded Method 30

## J

Job Detection

- TAG\_DETECTJOBSEP 81
- TAG\_JOBSEPDETECTED 81

JPEG File Format 44

## K

KIPP Installation 7

Kofax Installation 7

## L

Licensing

Active Property 21

Linking Controls

ImageDataChanged Event 2

ImageDataSource Property 1, 15

Load Time Improvement

Active Property 21

Loading Scanner Configuration

RestoreTagSettings Method 13, 33

Loading Scanner Driver

LoadScanner Method 9, 10

LoadScanner Method 9, 10, 31

LZW Compression 45

## M

Manual Timeout

TAG\_MANUALTIMEOUT 75

MDI Child Control

DynamicControl Property 23

Memory Requirements 5

Methods

AboutBox 21

Clear 22

ClearScanAhead 19, 22

ConfigureScanner 23

GetSavedScanner 9, 10, 24

GetTagChoiceCount 24

GetTagChoiceFlags 24

GetTagChoiceLong 25

GetTagChoiceRational 26

GetTagChoiceString 26

GetTagLength 12, 27

GetTagLengthDefault 27

GetTagLong 12, 27

GetTagLongDefault 12, 28

GetTagRational 12, 28

GetTagRationalDefault 28

GetTagString 13, 29

GetTagStringDefault 29

GetTagType 12, 29, 51

IsPageLoaded 30

LoadScanner 9, 10, 31

RestoreTagSettings 13, 33

SaveTagSettings 13, 34

ScanBatch 18, 35

ScanPage 17, 36

SelectScanner 36

SetTagDefault 12, 37

- SetTagLong 12, 37
- SetTagRational 37
- SetTagRational2 38
- SetTagString 39
- SetupScanner 11, 39
- UnloadScanner 9, 10, 39
- Most recently used scanner 9, 10, 24
- Multi-Page Files 17
  - OutputFileAppend 16

## O

- Options, Enabling 9
- OutputFileAppend Property 16, 17, 31
- OutputFileFormat Property 16, 31
- OutputFileName Property 16, 32
- OutputToFile Property 16, 33

## P

- Page Detection
  - IsPageLoaded Method 30
- Page Size Detection 58
- Page Size Tag 58
- Page Size, Default 9
- PageIndex Parameter 20
- PCX / DCX File Format 44
- Pixel Translations Technologies 40
- PowerTools Installation 8
- Programming Considerations 5
- Properties
  - Active 21
  - DynamicControl 23
  - ImageDataSource 1
  - OutputFileAppend 16, 17, 31
  - OutputFileFormat 16, 31
  - OutputFileName 16, 32
  - OutputToFile 16, 33
  - ScanAhead 34
  - Scanner 9, 10, 35
  - UseISISPipe 40
  - UseScannerCompression 13, 40

## R

- Reading Default
  - GetTagRationalDefault Method 12
- Reading Tag Defaults
  - GetTagLongDefault Method 12
- Reading Tags

- GetTagLong Method 12
- GetTagRational Method 12
- GetTagString Method 13
- Resolution
  - TAG\_RESUNIT 57
  - TAG\_XRESOLUTION 56
  - TAG\_YRESOLUTION 57
- RestoreTagSettings Method 13, 33
- RLE Compression 6, 43, 46
- Run Length Encoding Compression 6, 43, 46
- Runtime Licensing 4

## S

- SaveTagSettings Method 13, 34
- Saving Scanner Configuration
  - SaveTagSettings Method 13, 34
- Scan Ahead
  - TAG\_SCANAHEAD 76
  - TAG\_SCANAHEAD\_MAXPAGES 77
  - TAG\_SCANAHEAD\_PAGES 77
- Scan Ahead Pages 19
- Scan Area
  - TAG\_IMAGELENGTH 59
  - TAG\_IMAGEWIDTH 60
  - TAG\_XPOSITION 59
  - TAG\_YPOSITION 59
- Scan Destination 15
  - OutputToFile Property 16
- ScanAhead Property 34
- ScanBatch Method 18, 35
- Scanner Compression
  - UseScannerCompression Property 13, 40
- Scanner Configuration 23
- Scanner Control Tags 11
- Scanner Driver, Unloading 39
- Scanner Property 9, 10, 35
- Scanner Selection 36
- Scanner Selection -- GUI 8
- Scanner Selection -- Programmatic 9
- Scanner Setup 39
- Scanner Setup -- GUI 11
- Scanner Setup -- Programmatic 12
- Scanner Speed 6
- Scanning Events
  - DidScan 20, 23
  - WillScan 20
- Scanning to File 33
- ScanPage Method 17, 36
- ScanTags 11
- SCSI Controllers 7

- Selecting Scanner
  - LoadScanner Method 31
- Selecting Scanner Driver
  - Scanner Property 9, 10
- SelectScanner Method 36
- SETSCAN.INI 9
- SetTagDefault Method 12, 37
- SetTagLong Method 12, 37
- SetTagRational Method 37
- SetTagRational2 Method 38
- SetTagString Method 39
- Setting Tags
  - SetTagLong Method 12
- Setup of Hardware 7
- Setup... Button 9
- SetupScanner Method 11, 39
- Single Page Scanning 36
- System Requirements of Scanning 5
- System-level Drivers 7

## T

- Tag Inquiry
  - GetTagChoiceCount Method 24
  - GetTagChoiceFlags Method 24
  - GetTagChoiceLong Method 25
  - GetTagChoiceRational Method 26
  - GetTagChoiceString Method 26
  - GetTagLength Method 27
  - GetTagLengthDefault Method 27
  - GetTagLong Method 27
  - GetTagLongDefault Method 28
  - GetTagRational Method 28
  - GetTagRationalDefault Method 28
  - GetTagString Method 29
  - GetTagStringDefault Method 29
  - GetTagType Method 29, 51
- Tag Setting
  - SetTagDefault Method 37
  - SetTagLong Method 37
  - SetTagRational Method 37
  - SetTagRational2 Method 38
  - SetTagString Method 39
- TAG\_ADFTIMEOUT 75
- TAG\_BACKOFF\_200 90
- TAG\_BACKOFF\_300 90
- TAG\_BACKOFF\_400 90
- TAG\_BAR\_AUTOREADBACK 98
- TAG\_BAR\_AUTOREADFRONT 97
- TAG\_BAR\_MAXCOUNT 97
- TAG\_BAR\_MAXRATIO 97

- TAG\_BAR\_MINCOUNT 97
- TAG\_BAR\_MINHEIGHT 95
- TAG\_BAR\_SEARCHCOUNT 96
- TAG\_BAR\_SEARCHMODE 96
- TAG\_BAR\_SEARCHTIME 96
- TAG\_BAR\_TYPE 95
- TAG\_BAR\_TYPECOUNT 95
- TAG\_BARCODE 94
- TAG\_BARDATA\_LENGTH 99
- TAG\_BARDATA\_ORIENTATION 99
- TAG\_BARDATA\_SEARCHTIME 100
- TAG\_BARDATA\_TEXT 99
- TAG\_BARDATA\_TYPE 98
- TAG\_BARDATA\_WIDTH 99
- TAG\_BARDATA\_XPOSITION 98
- TAG\_BARDATA\_YPOSITION 98
- TAG\_BELL 73
- TAG\_BITSPERSAMPLE 63
- TAG\_BLUEBRIGHTNESS 62
- TAG\_BRIGHTNESS 61
- TAG\_BUFSIZE 73
- TAG\_COLORMAP 71
- TAG\_COLORRESPONSECURVES 69
- TAG\_COLORRESPONSEUNIT 69
- TAG\_COMPRESSION 65
- TAG\_CONFIG\_1 77
- TAG\_CONFIG\_2 77
- TAG\_CONFIG\_n 78
- TAG\_CONFIGURATIONS 77
- TAG\_CONTRAST 61
- TAG\_DESTINATION 73
- TAG\_DETECTJOBSEP 81
- TAG\_DETECTPAGESIZE 58
- TAG\_DITHER 65
- TAG\_DROPOUT 80
- TAG\_DTCFILTER 83
- TAG\_DTCGRADATION 82
- TAG\_DTCSELECTION 81
- TAG\_DTCSMOOTHING 83
- TAG\_DTCVARIANCE 82
- TAG\_ELECENDORSER\_MODE 87
- TAG\_ELECENDORSER\_STRING 86
- TAG\_ELECENDORSER\_YOFFSET 87
- TAG\_ELECENDORSER\_ZOOM 87
- TAG\_EMPHASIS 79
- TAG\_ENDORSER\_INCSTART 84
- TAG\_ENDORSER\_STRING 84
- TAG\_ENDORSER\_YOFFSET 85
- TAG\_ENHANCE 89
- TAG\_FEEDER 75
- TAG\_FEEDEROFFSET 75

- TAG\_FILLORDER 67
- TAG\_FRONTOFF\_200 89
- TAG\_FRONTOFF\_300 89
- TAG\_FRONTOFF\_400 89
- TAG\_FRONT\_PANEL 78
- TAG\_GAMMA 63, 78
- TAG\_GRAYRESPONSECURVE 70
- TAG\_GRAYRESPONSEUNIT 70
- TAG\_GREENBRIGHTNESS 62
- TAG\_GREENCONTRAST 61
- TAG\_GROUP3OPTIONS 66
- TAG\_GROUP4OPTIONS 67
- TAG\_IMAGEDESCRIPTION 72
- TAG\_IMAGELENGTH 59
- TAG\_IMAGEWIDTH 60
- TAG\_JOBSEPDTECTED 81
- TAG\_KFACTOR 74
- TAG\_LAMPTIMER 74
- TAG\_MANUALTIMEOUT 75
- TAG\_MATRIX 88
- TAG\_MAXBUFSIZE 80
- TAG\_MAXPAGES 76
- TAG\_MAXTIME 73
- TAG\_MIRRORIMAGE 79
- TAG\_MIXEDSCAN 79
- TAG\_NEWSUBFILE 72
- TAG\_ORIENTATION 68
- TAG\_OUTLINE 80
- TAG\_PAGESIZE 58
- TAG\_PHOTOMETRICINTERPRETATION 65
- TAG\_PIXELPATCH 83
- TAG\_PLANARCONFIGURATION 67
- TAG\_PLATENLENGTH 74
- TAG\_PRIMARYCHROMATICITIES 71
- TAG\_REDBRIGHTNESS 62
- TAG\_REDCONTRAST 61
- TAG\_RESUNIT 57
- TAG\_ROWSPERSTRIP 68
- TAG\_SAMPLESPERPIXEL 64
- TAG\_SCAN\_AHEAD 76
- TAG\_SCAN\_AHEAD\_MAXPAGES 77
- TAG\_SCAN\_AHEAD\_PAGES 77
- TAG\_SCANNERID 74
- TAG\_SCANORIENTATION 60
- TAG\_SCANTYPE 76
- TAG\_SOFTWARE 72
- TAG\_SPAOFF\_200 86
- TAG\_SPAOFF\_300 86
- TAG\_SPAOFF\_400 86
- TAG\_SPASPEED\_200 85

- TAG\_SPASPEED\_300 85
- TAG\_SPASPEED\_400 85
- TAG\_STRIPBYTECOUNTS 68
- TAG\_STRIPOFFSETS 68
- TAG\_SUBWINDOW 92
- TAG\_SUBWINDOW\_COUNT\_BACK 93
- TAG\_SUBWINDOW\_COUNT\_FRONT 93
- TAG\_THRESHOLDTRACKING 82, 88
- TAG\_WHITEPOINT 70
- TAG\_WINDOW 90
- TAG\_WINDOW\_COUNT\_BACK 92
- TAG\_WINDOW\_COUNT\_FRONT 92
- TAG\_XBASEOFFSET 90
- TAG\_XPOSITION 59
- TAG\_XRESOLUTION 56
- TAG\_YPOSITION 59
- TAG\_YRESOLUTION 57
- Tags, Scanner Control 11
- Throughput vs. Scanner Speed 6
- TIFF File Format 45
- Timing of Licensing Verification
  - Active Property 21

## U

- Uncompressed Image Files 46
- Unloading Scanner Driver
  - UnloadScanner Method 9, 10
- UnloadScanner Method 9, 10, 39
- UseISIPipe Property 40
- UseScannerCompression Property 13, 40

## V

- Version Information
  - AboutBox Method 21

## W

- WillScan Event 18, 20, 41
- Window Support
  - TAG\_SUBWINDOW 92
  - TAG\_SUBWINDOW\_COUNT\_BACK 93
  - TAG\_SUBWINDOW\_COUNT\_FRONT 93
  - TAG\_WINDOW 90
  - TAG\_WINDOW\_COUNT\_BACK 92
  - TAG\_WINDOW\_COUNT\_FRONT 92
- Writing Files

OutputToFile Property 16

## X

Xionics Installation 8