Streamline Workflows using Barcodes

With the availability of fast, accurate barcode reading and writing toolkits, it's quite easy to include barcodes within your applications to reliably transfer information. With some basic information about barcode technologies, you can determine how their use could streamline your document-processing workflows, which symbology best suits your specific needs, and how to maximize their reliability. You can try out real barcodes using our web-based demo or downloadable SDKs, and see how easily barcodes can be applied to improve various document processing operations.

Why barcodes exist

When it comes right down to it, there's one main reason for barcodes: Machines have difficulty reading text. Although optical character recognition (OCR) has come a long way, the wide variety of possible fonts, damage caused by scanning and faxing, variations in viewing angles, and many other factors, will always make a string of characters more difficult to reliably decode than barcodes. The barcode standards impose strict guidelines on line widths, size, shape, encoding, error detection and correction, and other characteristics, all with the specific intent of reducing reading errors.

Happy machines make happy workflows

So, if the primary reason for using a barcode is to improve results of a machine reading it, how does that translate into more reliable workflows? Many characteristics of barcodes make them ideal for passing information via an image via faxing, scanning, and other operations. When data can be reliably obtained by automated processing, there is far less manual intervention required, greatly reducing the cost of importing that data. The following features set barcodes above plain text for automated data acquisition:

1. **Barcodes are resilient –** You can do a lot of damage to a barcode, caused by things such as faxing several times, or writing over it, yet still be able to read it accurately.

2. **Reading is very fast –** A top-quality product like Barcode Xpress can find all of the barcodes on a page in just a few milliseconds. Performing OCR on that same page can take hundreds of times as long…certainly up to several seconds or more. This difference adds up quickly when you're processing a large number of documents.

3. **Barcodes can heal** – Most 2D barcodes can survive much more than just general damage. Using built-in redundancy and error correction, many can be read even when some elements are completely missing. The Data Matrix code to the right, for example, decodes with 100% confidence even with a hole punched through it.

4. **Space-saving** – A typical 2D barcode can pack more information into the same space than the text it contains, while remaining highly readable.

5. **Easily found** – Barcode reading software can locate and decode multiple barcodes anywhere on a page, and in any orientation.

Common Barcode Uses

The general use case for barcodes is passage of data from images to databases. When applied to documents, barcodes can serve many applications based on this concept. Here are just a few of the many common ways that barcodes are used to improve workflows:

1. **Batch scanning –** The simplest of all barcodes is the "patch code" that has long been used as a document separator in batch scanning applications. More complex barcodes can encode ID numbers, client names, dates, document names, and other information. An operator or automated system can use this added information to identify or classify a scanned document. Often, a barcode is printed before scanning, and attached to the first page of each new document. The decoded data can be used after scanning for accurate indexing.

The lowly patch code can only hold one of six different values.

2. **Printed forms –** If you are designing a new form, and you want to instantly recognize it amongst any collection of documents when filled-out forms are returned to you, simply include a barcode identifying the form. Many government forms, business reply cards, time sheets, and market surveys use preprinted barcodes for this purpose.

3. **Web based data collection –** Whenever an end user enters field data onto a Web form or PDF form that will ultimately be printed out, there is an opportunity to create a barcode on-the-fly that includes key components of

the user data. After that form is printed, faxed, copied, or mailed, the image enables reliable capture of the data that the user entered.

4. **Data verification –** Barcodes can be used to increase the reliability of almost any process by adding a second check. For example, many hospitals now scan and cross-check patient wristbands, charts, and each medication dose as it is administered, eliminating errors.

Selecting the optimal barcode type

Obviously, if you don't have the luxury of creating the barcodes in your workflow, you'll have to accept the choice of the original system designer. If you are reading product codes, for example, you'll need to recognize standard UPC (in the United States) or EAN (in much of the rest of the world) codes. Many existing 2D solutions use PDF417 to meet their requirements. If you can choose whatever barcode will best meet your needs, however, the following guidelines can help you narrow down your choices.

Choosing a linear solution

If you have a reasonably small amount of data to encode, less than 20 or 30 characters, you may wish to pick a linear, or 1D, barcode. These have the slight advantage of being recognizable by everyone as a barcode. Other than that, in most situations a 2D solution would yield higher reliability while consuming a smaller area. The best overall 1D choice for most new applications is Code 128. It allows you to encode uppercase and lowercase alphabets, numeric characters, and most ASCII control characters. Compared to most other 1D barcodes, it is relatively compact, yet fairly reliable. Code 128 includes one check character, reducing the likelihood of a false reading. The varying bar widths and relative complexity of the start/stop sequences, compared with some 1D codes, also reduce the chances that other data or images on the page might be falsely interpreted as a barcode.

As a guideline, the desired height for a typical 1D barcode should be at least 15% of its overall width, or 0.25 inches (0.64 cm), whichever is greater. If you are controlling how the created barcode will be processed (scanned or faxed, for example), you will want to make certain that there will be, at the very least, three pixels representing the smallest line-width in the code. Based on the vertical lines that it takes to represent each character, this will require a minimum of about 33 pixels per character, plus the start, stop, and checksum groups, which account for about another 105 pixels. Using these guidelines, a 10-character Code 128 should be at least 435 pixels wide and 65 pixels high. Naturally, this is the resolution you'll need ***after*** all processing, and you'll always want to round up to be safe. Refer to my earlier paper, ***Using Barcodes in Documents – Best Practices*** for more information on optimal barcode generation.

While Code 128 barcodes include a check character, there is still a reasonable probability of reading it incorrectly. A single error in decoding would always be detected, but if there are two errors the probability is roughly one percent that it will not be discovered at all. This “false positive” result could cause erroneous data to be entered into your database without any warning. Also, none of the common 1D barcode types includes any error ***correction*** at all. Unless your data can be cross-checked against a highly reliable data source, you should consider a 2D barcode that nearly eliminates false positives and adds the ability to actually correct errors.

Two-dimensional choices

The key strengths of 2D barcode formats are more capacity and error correction. Whereas it’s uncommon to find applications of 1D barcodes containing over 30 characters, it can be practical to encode even 500 or more characters in some 2D barcodes, while it is ***possible*** to store more than twice that many.

The Data Matrix code to the right, for example, contains 560 alphanumeric characters, and can be accurately read after scanning this page at 200 dpi. This is a reasonable scanner resolution setting for reproducing text with good visual quality, but would be marginal for optimal OCR. Most people would be unwilling to use a barcode much larger than this on their pages, unless the primary purpose of the page is to pass electronic data. A sample Data Matrix barcode containing 2,046 characters at approximately the same density would take up about 3 inches by 3 inches, completely dominating any page layout.

A Data Matrix containing 560 characters, readable when scanned at 200 dpi.

Data Matrix, and other 2D formats, go far beyond the basic error detection of any linear barcode. Using Reed-Solomon encoding, redundancy is built into the code when it’s created. These extra characters can be used during the decoding process to reconstruct all of the data even when portions of the original image are completely lost or destroyed. Thus, unlike a linear barcode, where each character exists in only one section of the image, each character is distributed across the image, and can often be recovered using the remaining portions, if an area is lost. As a general rule, it is possible to retrieve all of the data when up to 20% of the original image is affected. That ratio can be increased further by forcing more rows and columns than the minimum necessary, thus creating more redundancy, during the creation of the barcode.

Although Data Matrix codes are usually seen in a square format, the standard also allows for several rectangular options. The barcode shown here, for example, contains 65 alpha-numeric characters, arranged into a 16 row-high by 48 column-wide rectangle. This format still contains valuable error detection and correction capabilities, but may be easier to insert into your form design. It could just as easily be placed vertically on the page.

Comparing the space required for the same data to be encoded in a Code 128, PDF417, or Data Matrix barcode of approximately the same readability, it can be seen that the Data Matrix barcode can be more easily integrated when designing a new form.

Code 128, PDF417, and Data Matrix representations of “**12345Abcde**”

Because the smallest squares in the Data Matrix code are still wider than the narrowest bars in the Code 128 barcode, it will be more reliable in most cases. In general, the greater the smallest dimension of the smallest element is, the more abuse it can stand. Both the PDF417 and Data Matrix codes include redundant information that may be applied to attempt to correct an erroneous character.

Barcode insecurity

Just because you and I may not be able to decode a barcode just by looking at it, don’t ever assume that a barcode can be used to secure or hide sensitive data. As our samples demonstrate, almost anyone can easily extract the contents of any standard barcode. If you’re providing some other method of encryption before encoding a barcode, it is only that encryption that provides any protection…you could just as well have printed that encoded message as text or as a string of hexadecimal digits. You should never put any information into a barcode that you would otherwise be unwilling to print directly on a form.

Comparing Barcodes

If you’re choosing a 1D barcode, there are few benefits to choosing another over the Code 128 recommended above. There may be more to consider among the 2D options, as each has slightly different strengths. If you're creating barcodes, you should know that Barcode Xpress includes barcode writing functions for 1D, PDF417, and Data Matrix. Note that the practical data limit for 2D barcodes is roughly 500-800 characters.

SUMMARY OF BARCODE FEATURES					
Barcode Type	**Style**	**Holds**	**Max. data**	**Error correction**	**Notes**
Code 128	Linear	All ASCII	< 30 typ.	Detection only	1 character checksum
Aztec	2D	Digits Alphanumeric Binary	3832 3067 1914	Adjustable, increases size	5% to 95% of data region can be used for error correction. Used mostly in the transportation industry.
Data Matrix	2D	Digits Alphanumeric Binary	3116 2335 1556	Default is about 30%; can increase by upsizing.	Resilient. Compact. Rectangular option.
PDF417	2D	Digits Alphanumeric Binary	2710 1850 1108	Adjustable; adds from 2 to 512 codewords	Larger than other 2Ds; bar width is small, but barcode locator is large.
QR Code®	2D	Digits Alphanumeric Binary Kanji	7089 4296 2953 1817	Four levels from about 7% to 30%	Name "QR Code" requires trademark acknowledgement to Denso Wave, Inc., but free to use.

Trying out Barcode Xpress

To see if your particular image can be decoded using Barcode Xpress, you can use the Web demo at: demos.accusoft.com/barcodexpressdemo. While this will give you quick results, it's possible you can perform various types of pre-processing on your images to further improve results on difficult images. To perform more advanced testing, try out barcode writing, and see how to build barcode processing into your own application, download the free trial of the Barcode Xpress SDK. There are SDK versions supporting .NET and ActiveX /COM development for 32 and 64-bit deployment, plus Java and Java Mobile Edition toolkits.

The .NET and ActiveX SDKs include a copy of the ImagXpress component, providing a wealth of features for manipulating images. The Barcode Xpress component can just as easily be integrated with the ImageGear imaging SDK, and a sample project (separate from the SDKs) is also available for download, along with a brief video showing how to build a simple ImageGear barcode reading program.

Although your application may include image acquisition, pre-processing, viewing, database storage, and many other features, the actual barcode reading portion is extremely simple. Declarations, image acquisition, and display or storage of results are not shown, as those are all in the sample code, and would depend on your specific application.

```
// An array holds the barcode formats to be recognized
BarcodeTypes = SetBarcodeType();

// The Result class will hold details of the returned barcode(s)
Accusoft.BarcodeXpressSdk.Result[] results;

// Set the barcode types for the component
barcodeXpress1.reader.BarcodeTypes = BarcodeTypes;

/* This default searches the entire image. Replace the
   zeroes to select a pre-defined area for barcode search  */
System.Drawing.Rectangle currentArea
              = new System.Drawing.Rectangle(0, 0, 0, 0);
barcodeXpress1.reader.Area = currentArea;

// Set up a bitmap to pass a viewed image to Barcode Xpress
DIB = (IntPtr)imageXView1.Image.ToHdib(false).ToInt32();

// The Analyze method searches the page and returns the Results
results = barcodeXpress1.reader.Analyze(DIB);
```

Conclusion

Barcodes can be used within almost any workflow that might benefit by acquiring data from an image. When using barcodes, it's important to know the characteristics of each type to optimize reliability. Third-party SDKs like Barcode Xpress make it very easy to incorporate barcode reading and writing in new applications. Considering their ease of use and the potential benefits, it's surprising that so many forms still appear ***without*** barcodes.

You can find Accusoft Pegasus product downloads and features at www.accusoft.com. Please contact info@accusoft.com for more information.

About Accusoft Pegasus

Founded in 1991 under the corporate name Pegasus Imaging, and headquartered in Tampa, Florida, Accusoft Pegasus is the largest source for imaging software development kits

(SDKs) and image viewers. Imaging technology solutions include barcode, compression, DICOM, editing, forms processing, OCR, PDF, scanning, video, and viewing. Technology is delivered for Microsoft .NET, ActiveX, Silverlight, AJAX, ASP.NET, Windows Workflow, and Java environments. Multiple 32-bit and 64-bit platforms are supported, including Windows, Windows Mobile, Linux, Sun Solaris, Mac OSX, and IBM AIX. Visit www.accusoft.com for more information.

About the Author

Paul B. Firth, Product Manager
Paul Firth has been helping to guide the overall product strategy for Accusoft Pegasus since 2005. In this role he works closely with the Sales and R&D teams to identify and satisfy product needs in the highly-dynamic software tools market. Prior to joining Accusoft Pegasus, he has led high-tech marketing teams and managed teams of both software and hardware developers. Paul holds a Masters in Business Administration from New York University, and a Bachelor of Science degree in Electrical Engineering from the University of Rochester.

Curious? Scan this at 200 dpi or more, then send the image to our web demo (Data Matrix).