Publication: IVD Technology
Date: September 2008

Customizing software to prevent unnecessary bar code read failures

Automated laboratory systems whose parameters users can control will perform better.

Niels Wartenberg

Laboratory automation systems are ultimately only as accurate as the data they receive. For this reason, it is important to make the bar code identification system of a laboratory instrument as robust as possible in order to prevent misidentifications and avoidable failures to read (i.e., no-reads) from occurring.

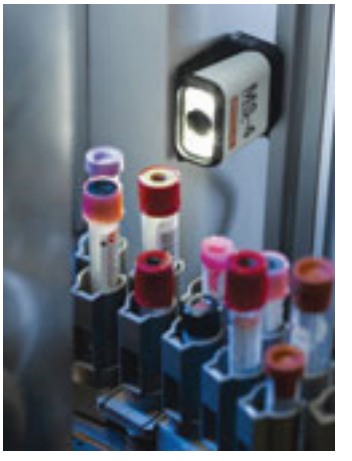

Figure 1. Developing effective interface software allows the user to customize its bar code system to meet application needs.

A significant number of the label symbols found in laboratories today either are of poor quality or do not meet the established industry specifications for symbols. Consequently, the system designer must consider how the automated system will respond to the poor-quality codes it encounters. Developing effective interface software that lets the user customize the bar code system to suit application needs is just as important as selecting the most appropriate bar code reading technology to embed inside the system (see Figure 1).

To create effective software, the system designer must first understand which software parameters the laboratory should be allowed to customize and which should be preprogrammed. Too much or too little laboratory control without a proper understanding of the subtle differences between symbologies can potentially increase the burden of service visits imposed on the system manufacturer, as well as result in an unnecessarily high number of no-reads.

This article can serve as a guide to help designers determine which options provide value to the user. Ultimately, the engineer needs to find the right balance between providing the user with enough flexibility and control over the application to maximize efficiency and building important safeguards into the system in order to ensure optimal performance and accuracy.

Application Challenges

Familiar point-of-sale (POS) applications can make bar coding seem very straightforward: Print a few bar codes, apply them to an item, and then scan them with a bar code reader. When analytical system designers look at how efficiently bar codes are read in POS applications, they may wonder why it is so difficult to achieve the same simplicity in the clinical laboratory environment.

The difference between the two applications largely comes down to bar code quality and the variety of bar code symbols. The retail industry has standardized on one symbol per market for all POS applications: the UPC—Universal Product Code—in North America, and its international counterparts elsewhere. High monetary penalties compel companies to comply with the industry's rigid symbol-quality guidelines. If a large grocery store chain receives goods from a manufacturer that do not meet the appropriate quality standards, it can fine the manufacturer heavily and threaten to discontinue carrying its product line.

The life science industry has established bar code quality standards as well. The Clinical Laboratories Standards Institute (CLSI) and the International Society of Blood Transfusion (ISBT) are among the associations that have specified a symbology or family of symbologies for specific applications.

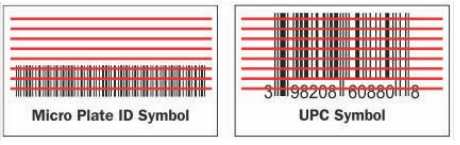

Figure 2. Its height is the built-in redundancy of a bar code. The UPC symbol provides much more redundancy than the narrow height of the Code 128 bar code that is due to the limited space available on microplates.

However, the guidelines are typically not followed or enforced to the degree they should be. Symbol quality in laboratory environments can vary dramatically from facility to facility, and even from day to day at the same facility. Either the symbols are produced out of specification originally, or the harsh environment to which they are subjected makes it difficult to maintain code quality. For example, condensation, stains, or marks can partially obscure the symbol. Extreme temperatures can cause the label media itself to deteriorate. If the symbols in these cases were to be scanned by a verifier, many would receive a solid F grade, failing the quality check with respect to multiple parameters (see Figure 2).

Likewise, because of the range of operations involved in the clinical diagnostic and drug discovery processes, one symbology is not suitable for every bar code application. A system's interface software therefore must be flexible enough to accommodate a variety of symbol combinations.

Until recently, linear symbols were used for nearly all applications. Many laboratories today, however, are introducing stacked and 2-D symbols. For example, where in the past a Code 39 or Code 128 symbol was applied to the narrow end of a microplate to identify the plate, today a Code 128 symbol may encode the plate ID, or that task may be handled by a PDF417 or a Data Matrix symbol. In many cases, the 96-well microtiter plate has been replaced with an array of 96 discrete vials with the same form factor as the microtiter plate. Each individual vial in such an array often is marked with a Data Matrix code (see Figure 3). Unlike equipment for POS applications, clinical laboratory instruments and the bar code readers inside them must be capable of accommodating these many variables at high speeds and with extremely high precision.

Figure 3. Laboratories use multiple symbols in combination to track specimens.

Recommended Software Parameters

To make the bar code system as efficient as possible, and to eliminate unnecessary no-reads, the interface software's parameter settings should take into consideration the potentially harsh and highly variable nature of the laboratory environment.

Figure 4. The quiet zone is the no-print area that borders the perimeter of the symbol.

A significant number of no-reads are caused by poor symbol quality. Several factors define a symbol of good quality: good contrast, proper dimensions, and sufficient quiet zone. The quiet zone, sometimes referred to as the no-print zone, is the space immediately surrounding the bar code (see Figure 4). Quiet-zone violations and under- or oversizing the printed elements of the symbol are by far the most common causes of no-reads (see Figure 5).

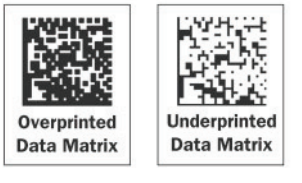

Figure 5. A significant number of no-reads are caused by over- or undersizing the elements of the symbol as printed.

Unwarranted no-reads can be avoided by allowing the laboratory to adjust certain software parameters to accommodate a batch of poor-quality codes. While it may seem advantageous for the manufacturer to limit the laboratory's control over bar code reader configuration parameters in order to ensure that its technical support staff knows what state the system is in, that approach actually increases the amount of on-site support needed over the long term.

For example, if an analyzer's bar code configuration software is programmed to accept only symbols of grade B quality or higher, and a significant number of the bar codes circulating in the laboratory are grade D or lower, the analyzer may encounter an unnecessarily high number of no-reads. The laboratory experiences a large amount of downtime because the

analyzer rejects the majority of its codes, and the manufacturer accrues the additional expense of a technically needless service visit to identify a problem caused merely by poor bar code quality. By building customizable options into the software parameters, however, the software designer can enable the system to accommodate typical bar code–related problems. The challenge lies in knowing how much flexibility can be allowed before the system comes into risk of compromising data integrity.The following guidelines should provide the instrument user with enough flexibility to customize the bar code reader configuration so as to meet the demands of the laboratory's particular bar code application without allowing the user to impair the data integrity of the entire system.

Basic Settings. The software parameters can be divided into two levels: basic and advanced. Basic settings consist of two buttons for each symbology supported by the system. The first button allows the user to enable the symbology in the system. The second button provides access to the advanced settings, with which the user can customize the software parameters for that particular symbology.

While many symbologies are in use today, systems designed for clinical laboratory applications should support a minimum of four symbologies—specifically, Code 128, Code 39, Interleaved 2 of 5, and Codabar. In addition to these four, systems designed for drug discovery applications should support Data Matrix and PDF417. By supporting these symbologies, the system will accommodate legacy bar codes still in use while at the same time being ready for the increased use of 2-D codes in the future.

Advanced Settings. Advanced settings should provide the user with more control over the application and allow for specification of the type of symbol the system should look for. Some settings will be unique to a specific symbology, while other parameters apply to all symbologies. These settings include a button to require a fixed number of characters (or, for Data Matrix, a fixed symbol size), a field in which to enter the specified number of characters, and a button to allow or disallow reduced quiet zones.

Advanced settings, as the term implies, are designed for the user who is experienced with bar code applications. The system designer may find it prudent to provide laboratories with the option to password-protect the advanced settings as an additional precaution.

Fixed Character Number or Symbol Size. Each symbology should have a button by which the user can specify that the number of characters be fixed. An additional field, hidden or grayed out unless "Fixed Number of Characters" is enabled, should be available for entering the specified number of characters. This parameter acts as a filter, enabling the system to eliminate unnecessary information from the data output.

Since Data Matrix symbols are structured differently from bar codes, the advanced settings for Data Matrix should include a button to enable "Fixed Symbol Size" instead of "Fixed Number of Characters." This option allows the user to instruct the reader to ignore one symbol and read the other if two Data Matrix symbols of different sizes are placed on the same plate or otherwise appear in the same field of view. Once the "Fixed Symbol Size" option is selected, a drop-down menu should become available to provide a list of selectable predetermined values. The unit measurement for a Data Matrix code is the mil, as with bar codes. These symbols have an x-mil size. The values offered for selection are at the designer's discretion, based on what can be accommodated within the minimum supported element size and maximum space available for the symbol. They might include the following:

8x8	12x12	16x16
8x18	12x26	16x36
8x32	12x36	16x48
10x110	14x14	18x18

Reduced Quiet Zones. Situations occur in which the reader is unable to decode a symbol because it cannot recognize the quiet zone required by the symbol specification. The two most common causes of this are obstructions and misprinted labels.

Obstructions typically are pieces of equipment, such as racks or robot-arm grippers. While these handling devices are taken into account during the design phase of the system, users might place the symbols slightly outside the target area defined by the designers and thus introduce the possibility of their being obscured.Misprinting of labels occurs when the media is not properly aligned in the printer. One side of the symbol will be closer to the edge of the label than allowed by the specification. Providing a "Reduced Quiet Zones" button enables the decoding of such symbols. This parameter configures the bar code reader to allow the decoding of symbols with quiet zones as small as approximately half the value required by the specification.

In the best system design, selecting the "Reduced Quiet Zones" button would trigger a pop-up window with a message stating that this change will apply to all enabled symbologies. Allowing reduced quiet zones should also automatically disable "Large Intercharacter Gap" for discrete symbologies. Enabling "Reduced Quiet Zones" or "Large Intercharacter Gap" independently of each other would not pose any risk to the data integrity of the system. The user should not, however, be able to activate "Reduced Quiet Zones" and "Large Intercharacter Gap" at the same time. While the chance that a large intercharacter gap will be misinterpreted by the reader as the end of the symbol is remote, preventing both options from being selected eliminates that risk.

Large Intercharacter Gaps. Discrete symbologies, such as Code 39 and Codabar, should include a "Large Intercharacter Gap" button.

Intercharacter gap is a term that refers to the data formatting of discrete symbologies. In a discrete symbology, each character begins and ends with a bar. The individual characters are separated by spaces that do not contain information. This space between characters is referred to as the intercharacter gap and is analogous to letterspacing in type (A B C D as opposed to ABCD). As a general rule, the intercharacter gap should not be greater than four times the narrow-bar width (Figure 6).

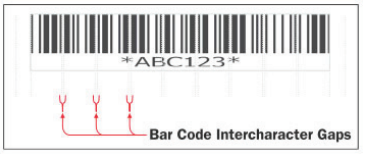

Figure 6. The intercharacter gap is to the space between characters in discrete symbologies.

Enabling "Large Intercharacter Gap" should cause the appearance of a pop-up window with the message that selection of this option will automatically disable "Reduced Quiet Zones" for all symbologies.

Additional parameters pertinent to specific symbologies also can be advisable. For example, the designer may consider adding a button under the Data Matrix symbology to enable or disable legacy Data Matrix symbols (those other than ECC 200). While other types of Data Matrix symbols do exist, only ECC 200 utilizes the Reed-Solomon error correction method. It is the only version suitable for new applications.

Scanners and No-Reads

While well-designed software allows the user to customize its application to make it as efficient as possible, the system will not be able to eliminate all no-reads related to poor symbol quality. No-reads quite often may seem to be caused by a misfunction in the hardware or system software when actually they are the result of a symbol problem. Consequently, it is important to understand how various bar code technologies decode symbols and how to most effectively employ them in an application.

Consider this typical case: A laboratory prints multiple sheets of bar codes and applies the codes to several hundred microplates. But each bar code has a significant quiet-zone violation on the right side. A laboratory technician scans the bar codes with a handheld scanner to verify that the codes are readable before loading the plates into the plate handler. The plate handler transports the microplates to the liquid handler where the bar codes are scanned again to log the process. However, when the plates arrive at the pipetting station, the scanner sends out a no-read message for each bar code in this batch because it recognizes the quiet-zone violation.

The root problem is that the batch of bar codes includes a quiet-zone violation that should be fixed. However,

from the laboratory's point of view, if one scanner can read the bar codes, then all the scanners should. Its conclusion is that there must be something wrong with the scanner at the pipette station, because the other scanners read the entire batch of bar codes without incident. In such a situation, the laboratory will inevitably call the manufacturer of the pipette station to report that something is wrong with its scanner. The manufacturer, at its expense, will send a service engineer out to look at the scanner that is actually functioning correctly.

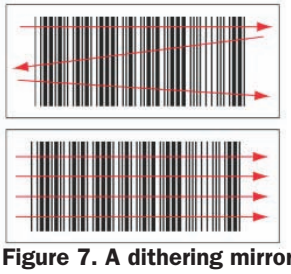

Figure 7. A dithering mirror moves the laser spot in a zigzag path across the bar code. A spinning polygon moves the laser spot across the symbol from left to right with no return path.

The first two scanners were able to read the bar code in spite of the significant quiet-zone violation because of either their orientation or the technology they employed to read the symbol. Handheld scanners, for example, are much more accommodating of one-sided quiet-zone violations because they use a dithering mirror. The laser spot makes a zigzag path across the bar code, reading it from left to right, then right to left on the return path (see Figure 7).

Stationary Scanners. Stationary scanners typically employ a spinning polygon (see Figure 7). To achieve the high scan rates necessary in automated applications, the laser spot moves across the bar code in one direction. Since there is no return path, whether or not the scanner can read the bar code with a quiet-zone violation will depend on which side of the code the quiet zone appears in relation to the orientation of the scanner. The stationary scanners mounted at the pipetting station and at the liquid handler in the case described were mounted in opposite orientations; therefore, one was able to read the symbol and one was not.

Imagers. If the scanners in the example had been imagers, the orientation of the unit would not have affected readability. Array imagers capture the entire symbol at once and process it two-dimensionally, as opposed to scanning the symbol one line at a time (see Figure 8). That makes the orientation of the symbol also irrelevant. In general, imagers are much more forgiving of low-contrast, damaged, or partially obscured symbols, all of which are commonly found in the laboratory environment (see Figure 9).

Figure 8. Imagers use an array of pixels to capture an entire symbol and process it two-dimensionally.

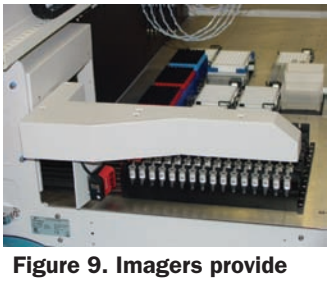

Figure 9. Imagers provide designers with an option that allows one unit to read both linear and 2-D symbols.

While a reliable scanning solution is important to building a robust bar code system, even the most efficient scanners cannot resolve all quality-related issues. Educating the user on the importance of producing good-quality symbols may go a long way toward preventing readability issues from occurring, which can save both the laboratory and its equipment suppliers time and financial resources in the long run. Of course, the nature of the laboratory environment inherently will make eliminating poor-quality symbols a difficult challenge.

System Tests

The interface software for the bar code solution is subjected to two types of system testing: design testing and manufacturing testing. Design testing is an R&D function intended to ensure that design goals are met. This includes testing for the extremes of allowable symbol placement, symbol resolution, and other variables. This testing should be done early in the design process. Manufacturing testing is performed during system production to verify basic operation of the bar code subsystem.

It may be tempting to use for tests symbols that have seen some use in real environments, but this makes the root cause of no-reads difficult to identify. Questions regarding how bad a bad symbol has to be and whether the system should be able to read a particular quality-degraded symbol make this approach challenging at best.

Testing with good symbols printed exactly to the outer limits of the specifications will involve the acquisition of

costly photographic symbols, but it will be repeatable and controlled. Parameters to be tested may include the minimum and maximum ratios for Code 39 and Codabar and the minimum and maximum element size for all symbologies. Because overprinting reduces the size of the spaces and underprinting reduces the size of the bars in the code, then if the spec for the minimum element size is 7.5 mil (0.0075 in.), the system should be able to read a 6-mil, or even a 5-mil, symbol. A universal standard has not yet been established for creating bar codes for testing purposes. Until one is created, a full-service label house that provides photographic labels will be able to produce a label that matches any label specification the system manufacturer desires.

Niels Wartenberg is senior field applications engineer at Microscan Systems Inc. (Renton, WA). He can be reached at nwartenberg
@microscan.com.

For proper evaluation of the performance of the bar code system, the testing process should involve scanning large volumes of symbols—a minimum of 100,000—in order to acquire an accurate impression of the system's failure rate. Reading accuracy in the clinical environment, where data accuracy is paramount, should be at rates of 99.9% and above. Ultimately, the goal is to engineer a system that can handle less-than-perfect bar codes so that, when the symbol quality starts to deteriorate, the laboratory does not suffer unnecessary downtime due to no-reads.

Niels Wartenberg is senior field applications engineer at Microscan Systems Inc. (Renton, WA). He can be reached at nwartenberg@microscan.com.

Conclusion

Though it may seem advantageous to a system manufacturer to limit the laboratory's control over bar code reader configuration parameters so that its technical support staff will know the state of the system, to do so actually increases the amount of on-site support needed over the life of the system. Providing access to basic software parameters for laboratories allows users to customize the software to best meet the needs of their particular bar code applications.

If functionality is removed from the software, then the laboratory may require a custom solution to be able to set up its application. Likewise, when settings are available, a help-desk engineer or specialist can talk the user through the solution on the phone, enabling the system supplier to avoid the expense of an on-site visit. Over the long term, this can result in considerable cost savings for the manufacturer.

MICROSCAN®