

BioRobotics

VERIFYING MICROPLATE BAR CODES

*Marc Goldstein
Beckman Coulter, Inc.*

This ASB is written in answer to the following request:

“I wish to read plate bar codes and compare them with a pre-defined worklist. The system should be stopped if the plate bar code read does not match the expected bar code due to be processed at that time. Can this can be done with an ‘If-Then’ statement and calling the plate name as a variable?”

Summary

Although reading plate bar codes for logging purposes is an easy function requiring just one step (using the Bar Code Reader step from the Devices Palette), setting up a method to compare bar codes and use a worklist does require some specialized knowledge. This document will describe how to perform these functions through a simple example of bar code matching and a more sophisticated example using a worklist.

System Requirements

These examples were written using Biomek FX 2.1. The functions and syntax described would be expected to work in FX 1.5 or greater. The system must have a bar code reader device associated with a bar code reader ALP. With minimal modification, these methods could be made to work with a bar code reader associated with a Stacker Carousel.

User Requirements

These examples were written such that users with a basic understanding of the Biomek FX software and use of variables would be able to understand, implement, and customize the methods for their own use.

Technical Background

Only two items of specialized knowledge are needed to work effectively with bar codes.

The first item is the method of looking up a bar code so that it may be compared with an expected value. Bar codes are “properties” that are associated with labware. If a piece of labware has its bar code read (at the bar code reader) or assigned as part of an Instrument Setup or Stacker Carousel Setup step, and the labware is subsequently moved to a new ALP location, the bar code property moves along with the plate. The bar code property may be determined using the following expression:

```
Properties(“BR1”).Barcode
```

This expression looks at the BR1 ALP, finds the labware there, and returns its Barcode property. For example, to look up the bar code of a plate at P5, the expression would be:

```
Properties(“P5”).Barcode
```

There are circumstances where the location to be checked might be specified by a variable such as Plate_Location, for example. Presuming the method contains a defined variable called Plate_Location that equates to a valid plate location, the following would be an acceptable expression:

```
Properties(Plate_Location).Barcode
```

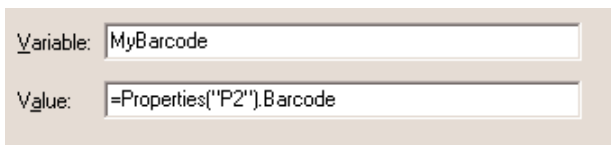
Notice that the explicitly defined ALP names have quotation marks around them, indicating that the names are specified as text strings. When using a variable to define the plate location, no quotes are

used. The lack of quotes indicates that Plate_Location is a variable and needs to be evaluated.

The above syntax will work with single labware (non-stacked) or the top piece of labware in a stack. To look up bar code information from labware in a stack, but not the top of the stack, a somewhat longer expression is needed that specifies the position in the stack, with 0 indicating the bottom position:

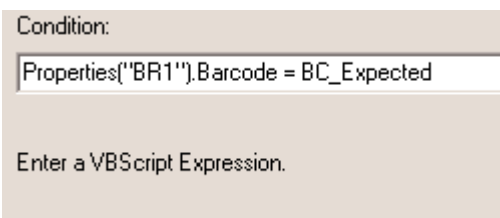
```
=World.Devices.Pipettor1.Deck.Positions("P6")  
.Stack(0).Properties.Barcode
```

In most circumstances, these expressions must be preceded by an equals sign to indicate that the whole Properties description is, in fact, an expression to be evaluated. To set a global variable called MyBarcode to equal the bar code of the plate at ALP position P2, the step would be configured as follows:



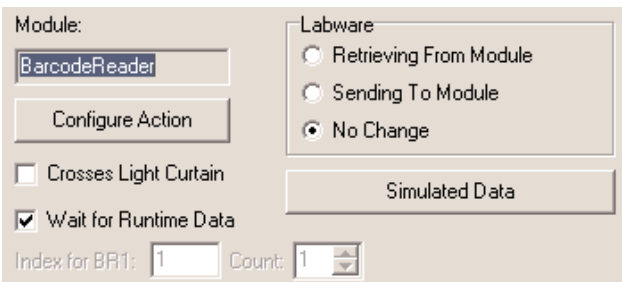
A configuration dialog box with two input fields. The first field is labeled "Variable:" and contains the text "MyBarcode". The second field is labeled "Value:" and contains the text "=Properties('P2').Barcode".

However, when using If-Then expressions, the equals sign is not needed because If-Then steps automatically evaluate the condition assuming it to be a Visual Basic-style expression. As a result, if we had a variable called BC_Expected that held the expected bar code, and we wished to compare that with the actual bar code of the plate at position BR1, we would set up an If-Then step in the following manner:



A configuration dialog box for an If-Then step. The "Condition:" field contains the text "Properties('BR1').Barcode = BC_Expected". Below the field is the text "Enter a VBScript Expression."

The next thing to remember when doing on-the-fly comparisons is that the Barcode Reader module needs to be configured properly. Specifically, the "Wait for Runtime Data" *must* be checked:



A configuration dialog for the BarcodeReader module. The "Module:" field is set to "BarcodeReader". The "Labware:" section has three radio buttons: "Retrieving From Module", "Sending To Module", and "No Change", with "No Change" selected. There is a "Simulated Data" button. The "Crosses Light Curtain" checkbox is unchecked, and the "Wait for Runtime Data" checkbox is checked. At the bottom, there are input fields for "Index for BR1:" (set to 1) and "Count:" (set to 1).

The full reasoning behind this is beyond the scope of this ASB, but a brief discussion follows.

Runtime Data and Enqueuing

The Biomek FX Software contains an "enqueuing engine" that processes the method steps and arranges the order in which they will execute. Each step broken down into component actions that are evaluated to see what resources are required to complete them. Examples of resources would be the Pods, ALP locations, devices, and labware. Under normal circumstances, each step is enqueued to execute at the earliest time that all needed resources are available. This allows methods using multiple Pods to execute efficiently, allows other things to go on during labware pauses or timed vacuum steps, etc. By clicking on the "Wait for Runtime Data" box in the Barcode Reader module, we tell the enqueuing engine that it should wait until that data is available before processing any later steps such as the If-Then step that will depend upon the data from the Barcode Reader module. This is necessary so that the If-Then step will be properly evaluated at runtime.

Creating a Simple Method: Example 1

The information above can be used in a small method that will move a bar coded plate to the reader, read the bar code, inform the user if the bar code matches a user-defined expected bar code, and then move the plate back to its starting location (see Figure 1).

Method Notes Step by Step

Instrument Setup

The deck used for this method is posted at the same location as the method. However, the method was written such that it should be able to be mapped to any deck containing P2 and BR1 locations.

A careful review of the method will show that an "actual bar code" is set in the Instrument Setup step. This is needed only when running the method in simulation without an instrument. During simulated runs, the Barcode Reader module will not change the effective bar code of the plate being read. Thus, for anything interesting to happen during a simulated run, a value needs to be set for the bar code.

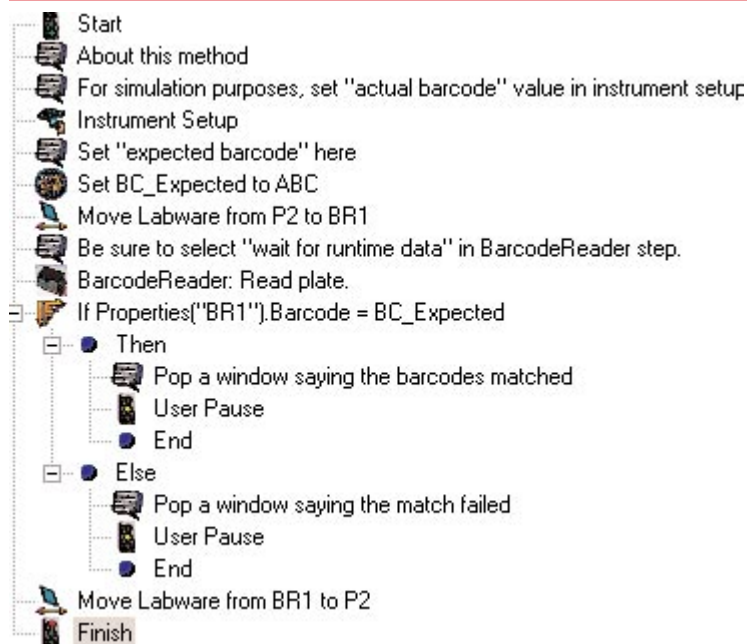


Figure 1.

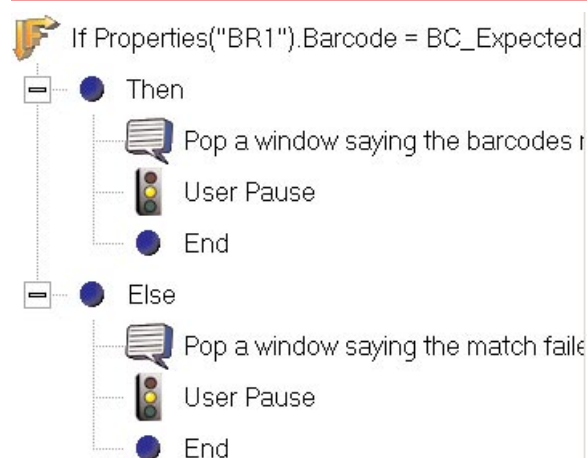


Figure 2.



Set BC_Expected to ABC

This step creates a variable called BC_Expected that contains the text ABC.



Move Labware from P2 to BR1

It is important to move the plate to the bar code reader prior to issuing the Read Plate command.



BarcodeReader: Read plate.

As described previously, this must be configured with the Wait for Runtime Data box checked.

The If – Then step (see Figure 2) is the heart of this method. It compares the actual bar code of the plate at BR1 with the expected bar code set by the user, provided in the BC_Expected variable.

If they match, a window will open stating that the match was confirmed. If not, a different window will inform the user that the match did not succeed.



Move Labware from BR1 to P2

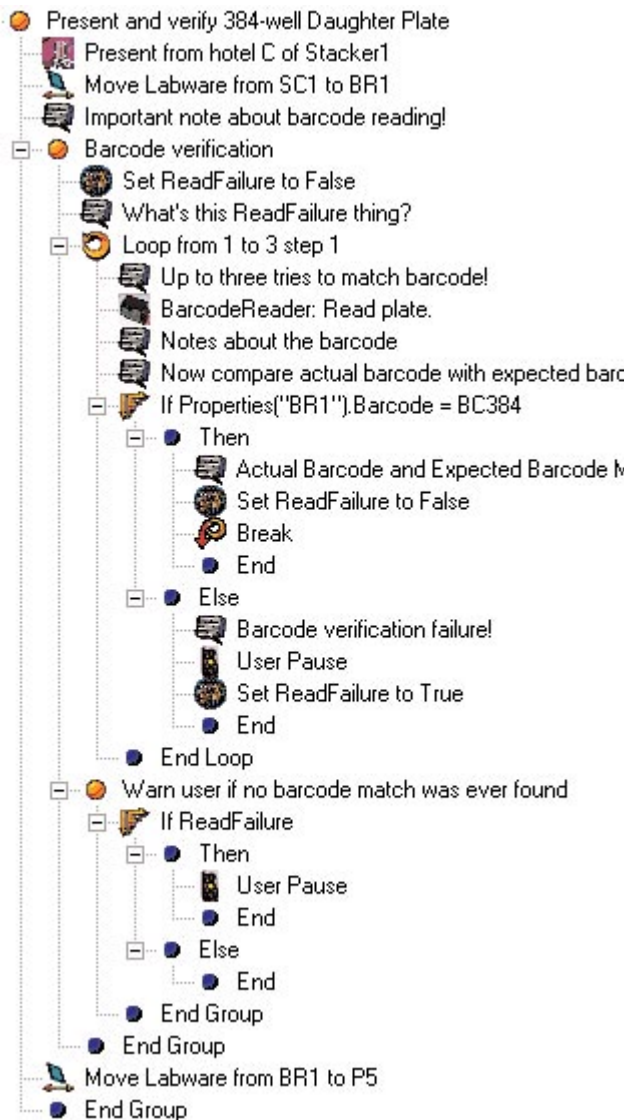
The final step in the method simply replaces the microplate on its original ALP location, P2.

Worklisted Bar Code Verification: Example 2

A common application for bar code verification is performing mother – daughter transfers. Example 2 will illustrate how bar code verification could work for 4 96-well mother plates being used to supply sample to quadrants of a 384-well daughter plate, with a worklist supplying all 5 bar codes for each family of plates.

This example contains steps that will alert the user to a bar code verification failure and provide an opportunity to investigate and correct the problem. This is performed using loops and the Break command which was instituted in FX 2.0. This method will not work on any software version before 2.0.

This method assumes that a stacker carousel will supply the plates and tips, as a good example of the integration of all the functions that would be needed to build this sort of a program. With some modifications it could be adjusted to work using only deck positions.



Each “Present and Verify...” group step contains a “Barcode verification” group step that handles checking the actual bar code against the appropriate variable read in from the worklist.

This group begins with the setting of a variable called ReadFailure to False. In the event of a failure to read the bar code, this variable will be set to True.

The main verification process is looped 3 times. This gives 3 attempts to read the bar code. After each attempt, using the BarcodeReader module, there is an If-Then step which compares the actual bar code of the plate to the expected one. If a match is found, the False value of ReadFailure is confirmed and the method breaks out of its loop, ending the bar code verification process. “Break” is an important new function in FX 2.0 and higher, and can be used in the development of complex methods.

If the bar code does not match the expected value, a User Pause step alerts the user that there has been a problem and instructs the user to correct it. The loop then iterates again, with another bar code read and checking of the value.

After the loop is concluded, the value of ReadFailure is checked and if it is True, a User Pause step alerts the user to the fact that no match was ever found. The method then continues.

The same Barcode Verification step was reused for each plate brought on, but modified so that the plates would be verified against the correct variable for 384 plate, quadrant 1 mother, etc.

Conclusion

Bar code verification with the ability to act upon failure is an important capability set. The flexibility of the Biomek FX software allows the user to customize how the instrument handles this type of situation. Several simple commands can be put together to provide sophisticated process control and error handling. This ASB provides samples of this functionality that may be customized by the end user to perform any number of tasks.

Limitations of Liability

Except as provided in writing signed by an officer to Beckman Coulter, Inc. (BCI), this software and any related documentation are provided "as is" without warranty of any kind, expressed or implied, including that the software is "error free." This information is presented in good faith, but BCI does not warrant, guarantee, or make any representations regarding the use or the results of the use of this software and related documentation in terms of correctness, accuracy, reliability, currentness, omissions, or otherwise. The entire risk as to the use, results, and performance of this software and related documentation is assumed by the user.

Except as expressly provided herein, BCI makes no other warranty, whether oral or written, expressed or implied, as to any matter whatsoever, including but not limited to those concerning merchantability and fitness for a particular purpose, nor is freedom from any patent owned by BCI or by others to be inferred.

BCI shall not be liable, to any extent whatsoever, for any damages resulting from or arising out of the use or performance of this software and related documentation or the procedures specified in the documentation for this product, regardless of foreseeability or the form of action, whether in contract, tort (including negligence), breach of warranty, strict liability or otherwise, and including but not limited to damages resulting from loss of data, loss of anticipated profits, or any special, indirect, incidental or consequential damages. In no event shall BCI's liability to the user exceed the amount paid by the user to BCI hereunder. The user assumes full responsibility for the results obtained from the use of this software and related documentation and for application of such results.

* All trademarks are the property of their respective owners.

* All trademarks are the property of their respective owners.



Developing innovative solutions in genetic analysis, drug discovery, and instrument systems.

Innovate **Automate**
SIMPLIFY

Beckman Coulter, Inc. • 4300 N. Harbor Boulevard, Box 3100 • Fullerton, California 92834-3100

Sales: 1-800-742-2345 • Service: 1-800-551-1150 • Telex: 678413 • Fax: 1-800-643-4366 • www.beckmancoulter.com

Worldwide Bioresearch Division Offices:

Australia (61) 2 9844-6000 **Canada** (905) 819-1234 **China** (86) 10 6515 6028 **Eastern Europe, Middle East, Africa** (41) 22 994 07 07
France 01 49 90 90 00 **Germany** (89) 35870-0 **Hong Kong** (852) 2814 7431 / 2814 0481 **Italy** 02-953921 **Japan** 03-5404-8359
Mexico 525-605-77-70 **Netherlands** 0297-230630 **Singapore** (65) 339 3633 **South Africa** (27) 11-805-2014/5 **Spain** (34) 91 3836080
Sweden 08-564 85 900 **Switzerland** 0800 850 810 **Taiwan** (886) 2 2378 3456 **Turkey** 90 216 309 1900 **U.K.** 01494 441181 **U.S.A.** 1-800-742-2345