SoftLinx V Protocol Builder

User Manual V 5.0.48

Hudson

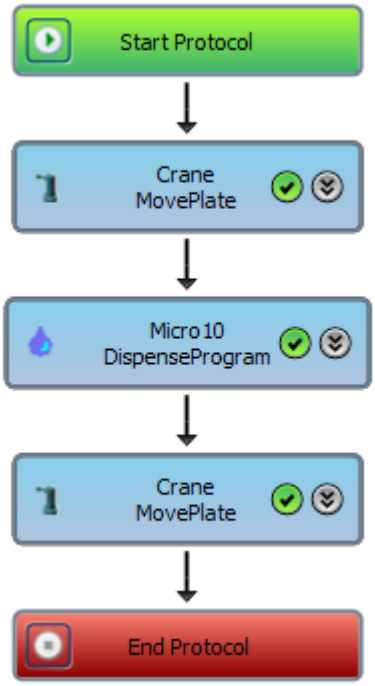
10 Stern Avenue, Springfield, NJ 07081

Tel: 973-376-7400 ♦ Fax: 973-376-8265

SoftLinx V Protocol Builder - Introduction

SoftLinx V is a multitasking laboratory control application, designed to program and operate laboratory workcells. Users build protocols which will define processes that instruments will execute, in some type of order.

What is a protocol?



Protocols are sets of instructions which the computer will send to instruments to execute. Protocols in this program are presented in a sequential flowchart format.

One simple example of a protocol is shown above. This particular protocol, when started, will move a plate using an instrument, named Crane, to the Micro10. Once that step is complete, the Micro10 will run a dispensing program on a plate. Once complete, the protocol will move the plate from the Micro10 to another location, again using the Crane.

Protocols can be very simple to extremely complex. There is no limit as to how many steps are in a single protocol.

Additional help can be found at http://www.HudsonRobotics.com, or by contacting Hudson at 973-376-7400.

Main Menu

Upon starting SoftLinx, or whenever no protocols are open, the main window will display the screen above. Users may do one of the following:

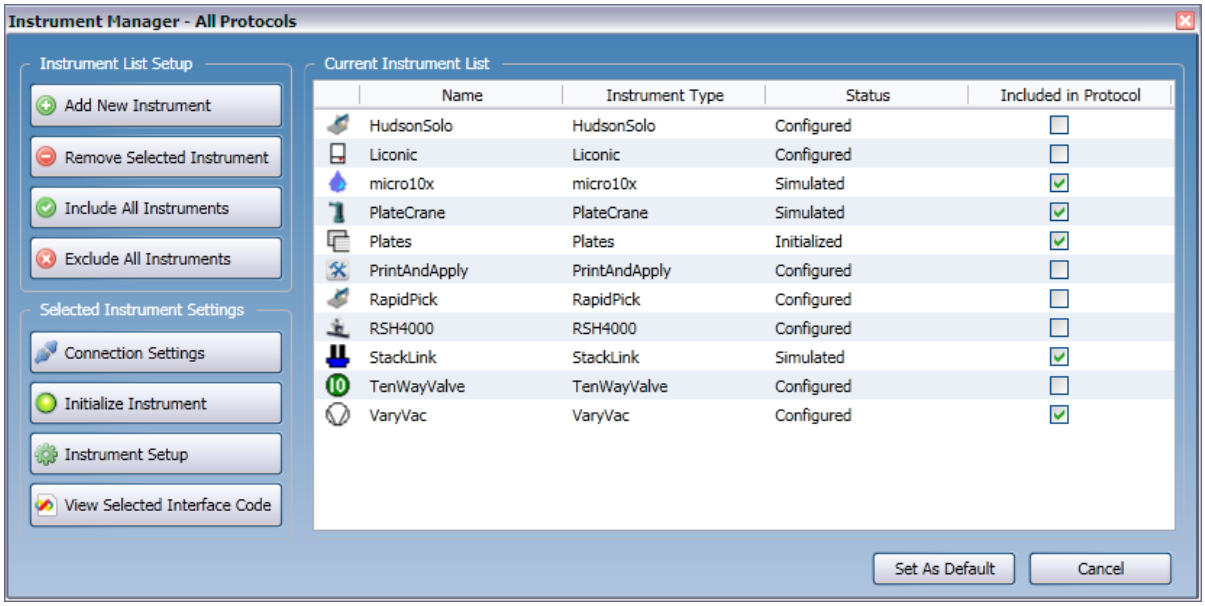
New Protocol - Creates a new protocol. Opens a new tab with an empty protocol.

Open Protocol - Opens a previously-saved protocol. Loads the protocol into a new tab.

Manage Instruments - Opens the Instrument Manager.

In addition, when a user clicks new protocol, or opens an existing protocol, the program will begin starting and initializing the default set of instruments, defined by Instrument Manager, to be used in protocols.

Instrument Manager



The instrument manager hosts a variety of options to set up the user's workcell in preparation for protocol development. The above list shows all instruments available to the user. Each instrument will list its unique name, an identification icon, its instrument type, its current interface status, and whether or not SoftLinx will open it up to be available in protocols. Although each interface's name must be unique, SoftLinx can simultaneously control multiple instruments of a single type.

If this window is opened at start up, it will set the instruments used for all new protocols to the specified configuration. If this manager is opened when a protocol is open, it will modify the available instruments for that specific protocol. The name of the modified protocol, or "All Protocols", will be specified in the title bar of this screen. Only instruments which are marked "Include in Protocol" will show up in the toolbox for that specific protocol. This allows SoftLinx to only open specific instruments when opening a protocol, and prevent unnecessary interfaces from loading to save computer memory.

Interfaces can have one of four statuses:

Configured - Connection settings have been established, but the interface has not been initialized. Initialization requires that the instrument must test its connection settings, to see if it is to be considered Initialized or Simulated.

Initialized - The interface has been connected to an active instrument, and can be used.

Simulated - The interface is active and can be used, but it does not have a connection to the instrument.

Not Configured - This instrument requires the use of the Connection Settings button.

Instruments which have been checked will be available for protocols to access.

In addition to the list of instruments, the following commands are available on this screen:

Add New Instrument - Installs instrument interfaces and identifies instruments which can be used in protocols. See the Adding Interfaces section.

Remove Selected Instrument - Remove the selected instrument from the list. In addition, if the instrument to be removed is the last instance of its instrument class within the list, the user also has the option to uninstall its associated interface. Protocols which require this removed instrument will prompt the user to reinstall the interface. In addition, all protocol steps which call this interface will be treated as inactive until the instrument is reinstalled.

Check/Uncheck All Instruments - All instruments in the list will be checked/unchecked. Instruments that are checked will be initialized by protocols, unchecked ones will not.

The following buttons will aid in instrument manipulation:

Connection Settings - Allows the user to modify communication parameters between the instrument and the computer.

Initialize Instrument - Applies the configuration settings to the instrument, and activates it for use. If successful, the instrument will be placed in an "Initialized" state. Instruments may also be put in a "Simulated" state if the instrument is unavailable.

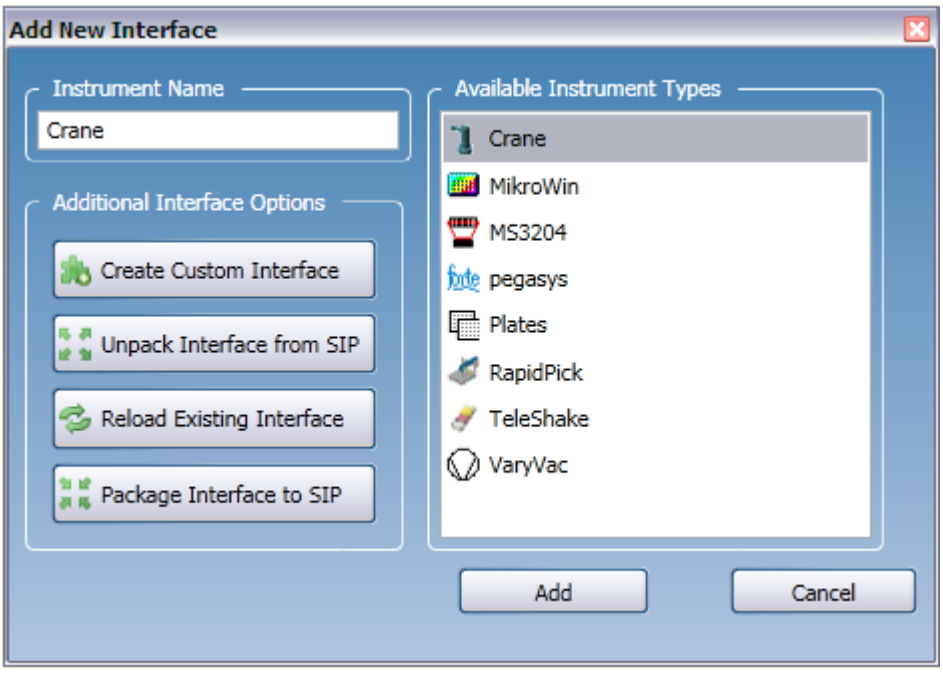
Instrument Setup - Calls the "Setup" command, opening the selected instrument's setup window to manipulate runtime settings, such as position setup, operation speed, and other runtime modifications.

View Selected Interface Code - Opens the VBA script for the selected instrument within its own VBA IDE (Integrated Development Environment). Users have the ability to debug and edit code of each interface.

Clicking cancel will undo all changes. However, it will not uninstall interfaces which have been recently installed.

Adding Interfaces

Clicking on the Add New Instrument button on the Instrument Setup list will activate the following window:

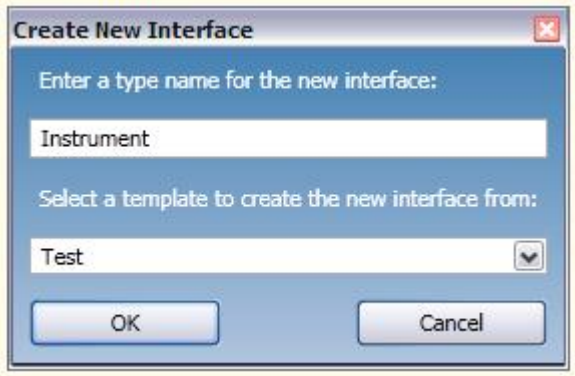


To add a new instrument, select an instrument type from the "Available Instrument Types" list. This is a list of all installed instrument interfaces, as well as pre-loaded SoftLinx Interface Packages (SIP), located in the Installed Directory\Interfaces folder. When clicking on an instrument in the list, its name will appear in the Instrument Name box. Users may customize names by typing in the Instrument Name box after selecting an instrument. Instrument Names can only be composed of alphanumeric characters and the underscore "_" character.

Interfaces which are already installed on the machine have their proper icon listed with their name. Interfaces which are not currently installed, but have an existing SIP on the system, are noted by the  icon.

In addition to installing interfaces, users may do the following:

Create Custom Interface - Allows the user to create a new instrument interface, based off of an available interface template. Clicking on the button will open the following:



Users name the interface by instrument type - not by instance name. Templates are also available to use as starting points for interface development. Once named, the newly created interface will immediately open and be available for development. In addition, the new interface is installed onto the machine, so users may access the instrument they are creating immediately. The interface is then added to the list.

Note that if the user uninstalls the interface without packaging it to a SIP, the interface cannot be recovered.

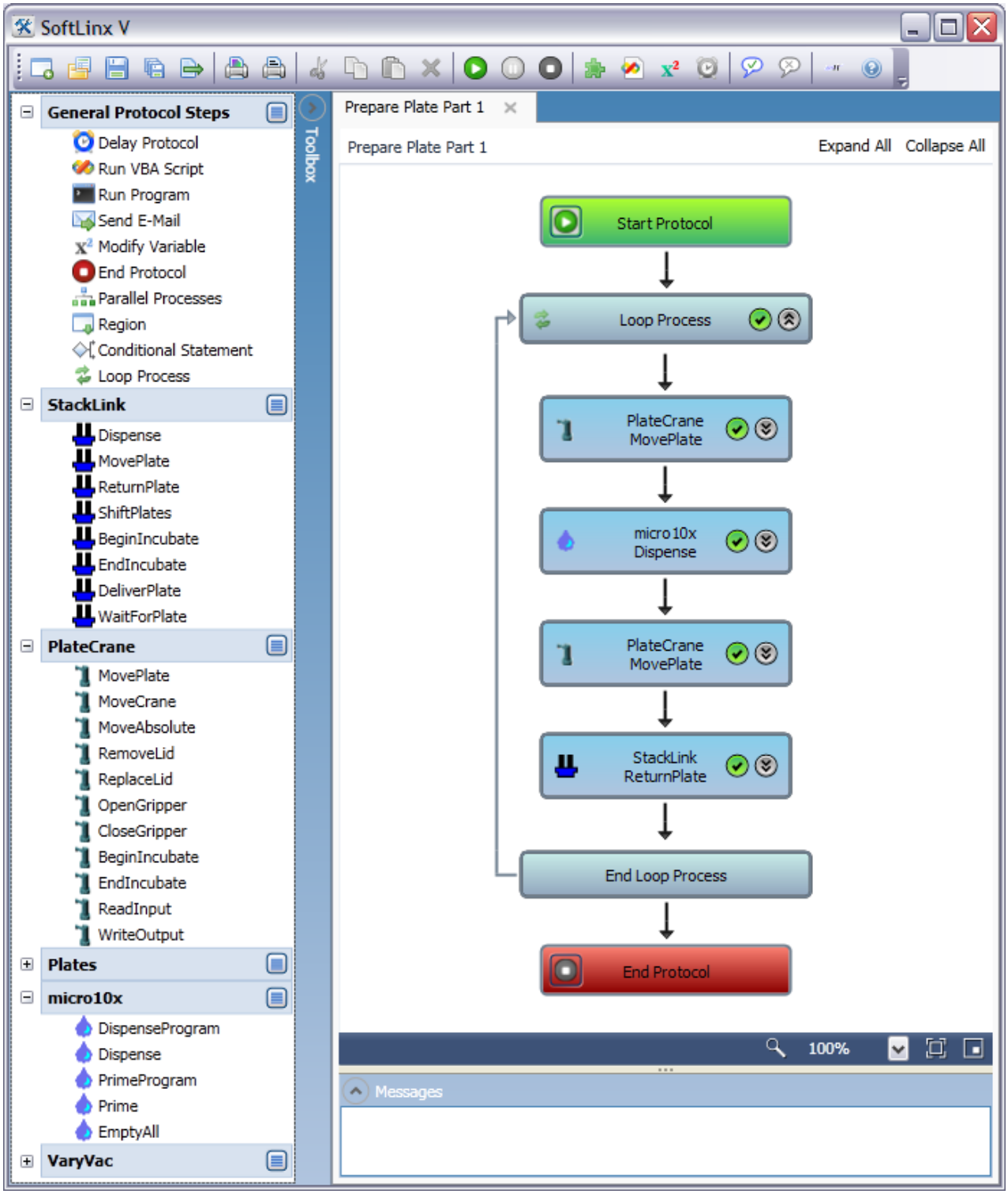
Unpack Interface from SIP - Installs an interface from a SIP, from a location defined by the user.

Reload Existing interface - Reinstalls an interface, and overwrites all installed files of an interface with those from the current associated SIP file.

Package folder to SIP file - Packages an Instrument folder to an SIP file, and saves it to a location defined by the user.

Clicking **OK** will install the selected instrument files to the computer, if that instrument's files are not installed. This will also add the custom-named instance of the selected instrument to the list of all instruments in the Instrument Setup screen. In addition, the interface will be activated to allow the user to set communication settings.

Protocol Editor Screen



The protocol editor window is the main SoftLinx V window. Protocols can be viewed, modified, and executed here. Multiple protocols can be opened and run simultaneously.

All available protocol steps are found in the toolbar on the left, and SoftLinx V system messages are reported in the Messages window.

Menu Bar:

The menu bar is located at the top of the window. This provides basic functions of the protocol editor window.

The following commands, with keyboard shortcuts, are accessible to users:

New Protocol - Opens a new tab with an empty protocol. (Ctrl + N)

Open Protocol - Opens a tab with a previously saved protocol. (Ctrl + O)

Save Protocol - Saves the currently viewed protocol to a .SLVP file. If it is a new protocol, the user must specify a save file name. (Ctrl + S)

Save Protocol As - Saves the currently viewed protocol to a new .SLVP file. The user must specify a save name. (Ctrl + Shift + S)

Export Protocol - Protocols can be exported as a text file or an image file for documentation purposes. Protocols exported as a text file (.txt) are saved in an outline format. For example, the protocol above is saved as the following file:

• Protocol: Prepare Plate Part 1

    • Loop steps 15 times.

        • Crane.MovePlate - Move Plate from 'Stack1' to 'Micro10x.Nest'

        • Micro10x.Dispense - Dispense 50uL to Plate.

        • Crane.MovePlate - Move Plate from 'Micro10x.Nest' to 'StackLink.Pos1'

        • StackLink.ReturnPlate - Return plate from 'StackLink.Pos1' to StackLink.Stack2

    • End Loop

• End Protocol

Protocols that are exported as an image can be saved under the following formats: BMP, GIF, JPG, PNG, TIF, and WPD.

Print to Image - Specify a printer and print the current protocol as an image.

Print to Text - Specify a printer and print the current protocol to a text file. The protocol will be printed in an outline format, just like the Export Protocol Function. (Ctrl + P)

Copy - Copies the selected steps to the clipboard. Does not remove the steps. (Ctrl + C)

Cut - Copies the selected steps to the clipboard. Removes the selected steps from the protocol, if such steps can be removed. (Ctrl + X)

Paste - Places the copied steps onto a selected location. (Ctrl + V)

Delete - Removes the selected steps from the protocol. (Delete key)

Manage Instruments - Opens the Instrument Manager. (Alt + F10)

Access VBA Code Script - Open the script of the VBA step in its own IDE.

(Alt + F11)

Manage Protocol Variables - Opens the Protocol Variable Manager.

Time Constraints - Opens the Time Constraint manager for a specific step.

Run Protocol - Starts execution of the currently visible protocol. (F5)

Pause Protocol - Pauses the currently visible protocol only. Pausing does not pause all protocols. Pressing Play resumes the protocol. (Space Bar)

Stop protocol - Stops and terminates the currently visible protocol, if it is running.

Show/Hide Protocol Comments - Expands/Hides all steps' comment boxes, which are located on all steps. Comment boxes allow users to add additional notes to each step.

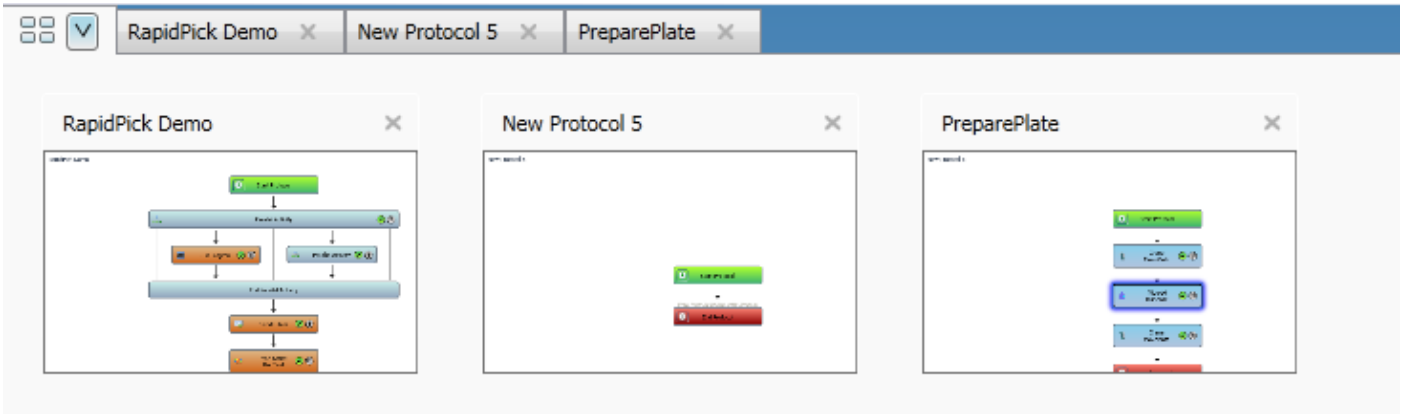
About - Shows information about this program, including version numbers of SoftLinx and all installed interfaces.

Help - Opens this help file. (F1)

Protocol Frame

The protocol designer frame shows the protocol currently being edited. Multiple protocols can be opened, but only one protocol can be shown at a time. Protocols are managed by a tab system, located at the top of the frame. Tabs can be selected to bring focus to that protocol. In addition, users can close protocols, or hover over tabs to see a preview of the underlying protocol.

When two or more protocols are active, an additional management tab is displayed to the left of all tabs. Users can click the tab to get a thumbnail view of all current protocols. Users can then select a protocol to focus on, or close other protocols.



The designer can also expand all or condense all protocol steps, by clicking on the Expand All or Collapse All links in the upper right corner of the frame. To restore protocol steps and allow for custom expand and collapse actions, click on the Restore link, which will replace the Expand or Collapse links when clicked.

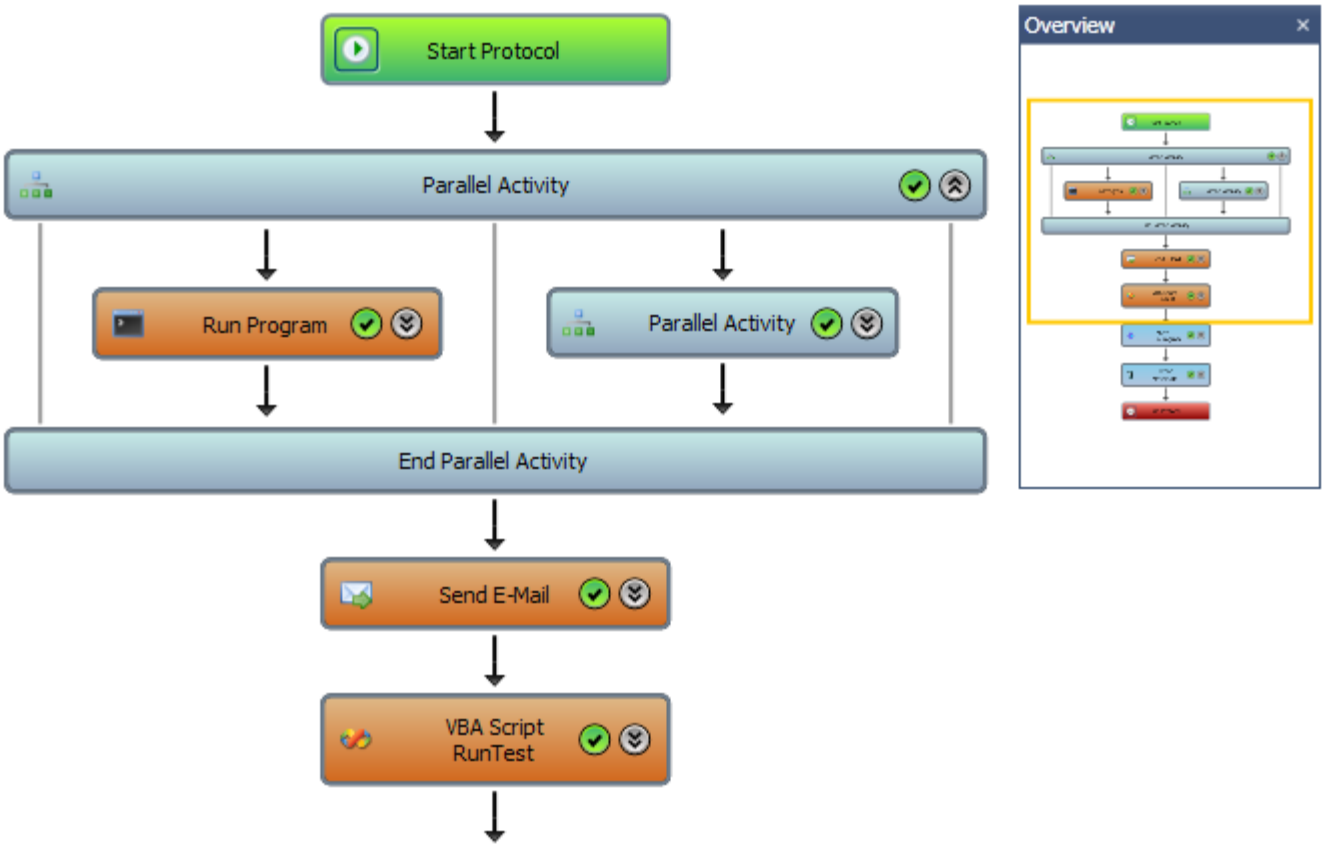
In the lower right corner of the frame are the protocol navigation tools. These are used to change the viewing area of the protocol.

The zoom value is customizable by the user, ranging from 25% to 400%.

The magnifying glass will restore the zoom value to a default of 100%.

The fit all button  will attempt to fit the entire protocol into the view window.

The mini map button  will create an overview window, or a mini map, of the entire protocol in a separate window. Users can drag the yellow box in the overview window to change the visible area of the protocol, as shown below.


Start Protocol
Parallel Activity
Run Program
Parallel Activity
End Parallel Activity
Send E-Mail
VBA Script
RunTest
Overview

Running the Protocol

Protocol execution is controlled by using the three buttons in the taskbar: 

These buttons will only control the currently visible protocol. Any other protocols running from other unselected tabs, or from other programs, will not be affected by these buttons.

The play button will only be active when the current protocol is stopped or paused. The pause button will only be active when the current protocol is running. The stop button will be active when the current protocol is running or paused. SoftLinx V protocols may run simultaneously.

Protocol Execution:

Once a protocol is saved and verified, users may execute the protocol. Users may either click Play on the taskbar or press F5.

Hitting the Run command will do the following, in order:

- Ask the user to save the protocol before execution, if the protocol has not been saved since the last change. If it has, it will confirm execution of this protocol. This is the only confirmation before execution.

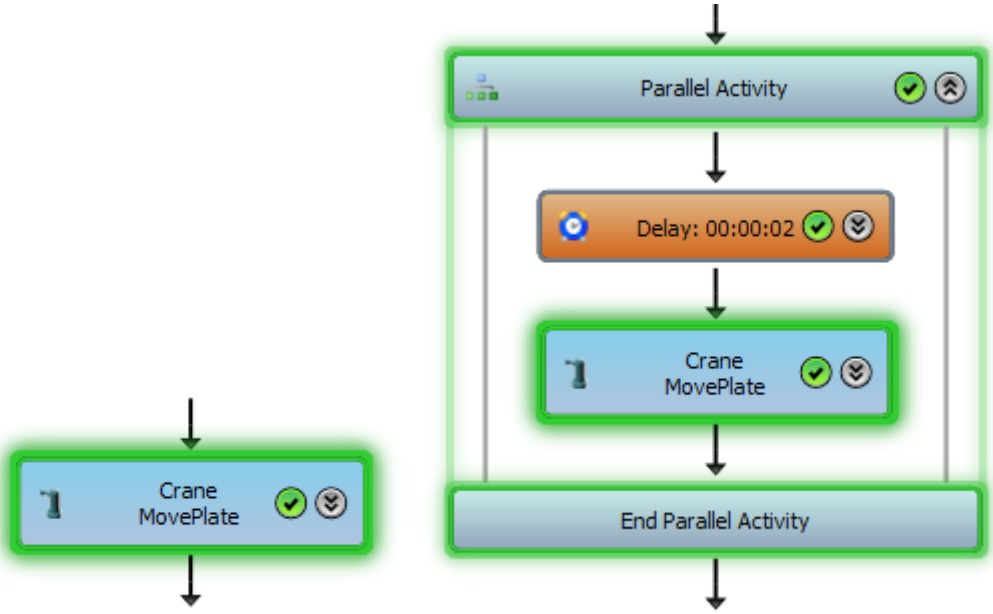
- Causes all active instruments, when not currently engaged in a protocol, to activate their startup procedures. Instruments do this one at a time, not simultaneously. Status messages will be displayed in the "Messages" box, located at the bottom of the main SoftLinx V window.

- The protocol will then begin execution. Steps will execute sequentially, from top to bottom.

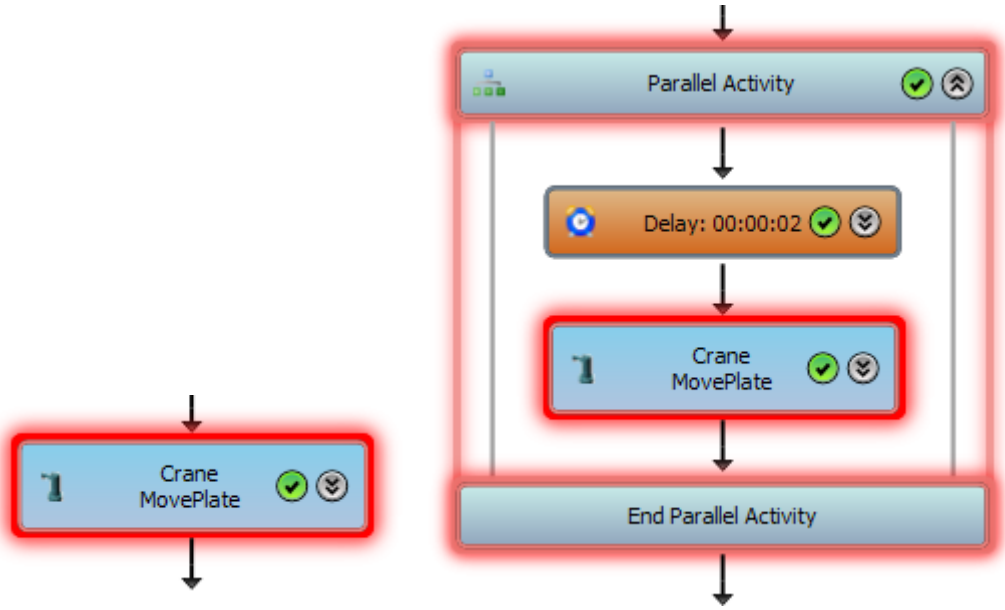
Runtime Options:

While a protocol is running, the user is able to monitor the runtime status of the overall protocol. Protocols can not be changed while executing. In addition, all protocol messages will be posted to the message log, located below the protocol frame. Execution of protocols can be interrupted with the pause or stop buttons.

Steps that are currently executing will be marked with a green aura. Expanded steps with children steps that are executing will be marked with a light green aura.



In addition, if a user manually stops a protocol run by SoftLinx, the last running steps which was executing will be marked with a red aura. Red auras will remain until the user restarts the protocol.

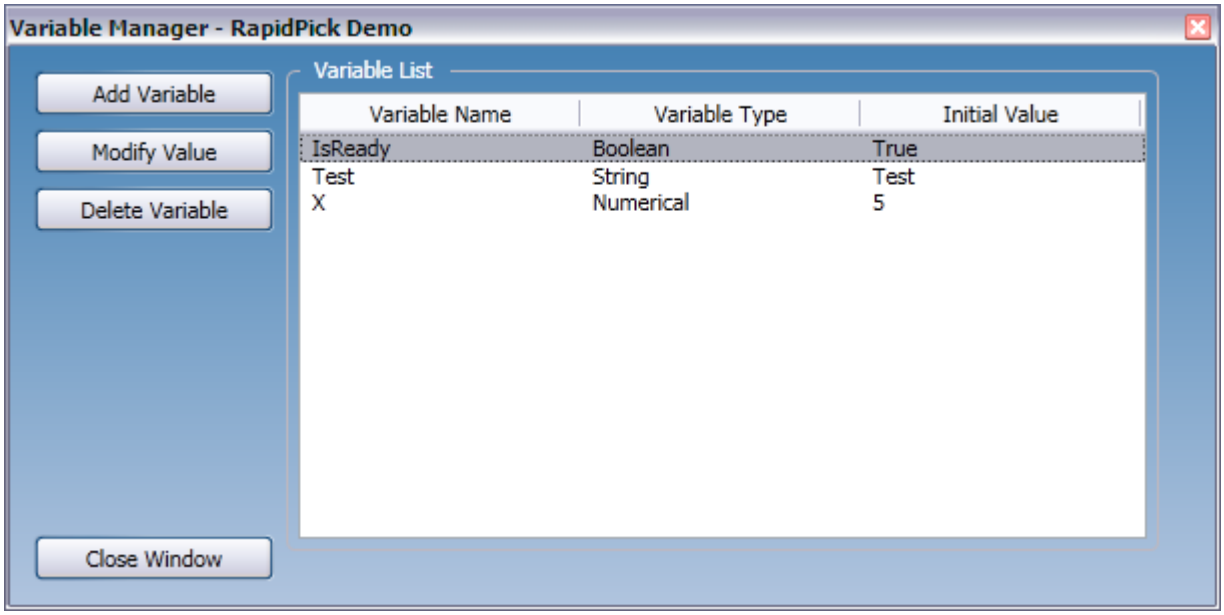


Protocol Variables

Protocol variables can be defined to help facilitate the execution of the protocol's steps. Variables are often used in controlling Loop Process steps, Conditional Statements, or even directly setting values within instruments themselves through the use of the Modify Variable step or the VBA Script steps.

By clicking on the variable icon, the user can define variables for that specific protocol. Variables defined in this manner do not transfer to other protocols outside of the one in which they are defined.

Protocols support True/False (Boolean) variables, numerical variables, and string variables.



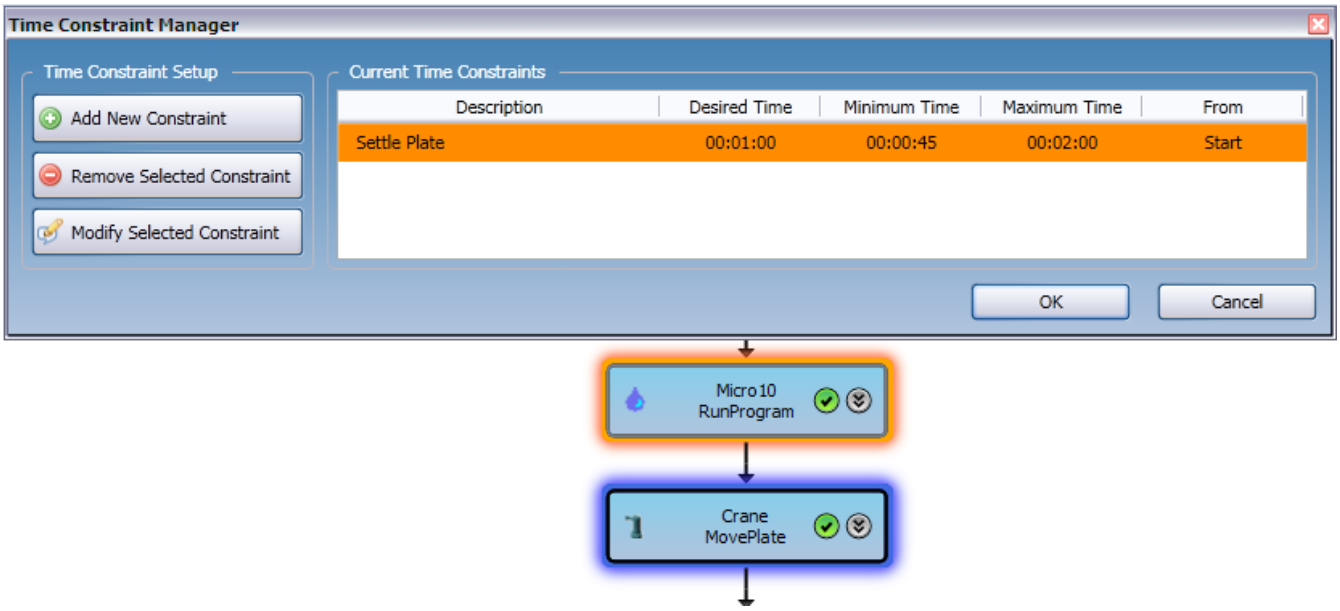
Variables must be named using alphanumeric characters and underscores only. Variable names must also begin with a letter.

Variables can be added, modified, or deleted from any protocol before execution.

Time Constraints

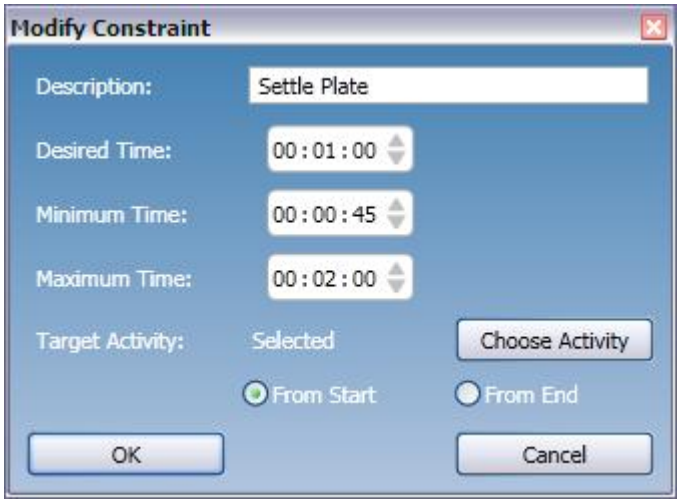
Time constraints are user-defined delays between two specific steps. This allows users to run a step at a specific time after a step has completed, for time sensitive actions. Time constraints do not need to be defined between two adjacent steps. However, time constraints can not be defined across loops.

Time constraints can be defined by clicking the time constraint icon on the main toolbar. Before opening the time constraint window, the step that is to be run second in a time constraint must be selected.



The time constraint manager shows all time constraint relations attached to the selected step. With the use of parallel and conditional steps, a single step can have multiple time constraints. Each constraint is shown with an orange aura.

When adding a new constraint, the user can define when a step is going to run, in relation to the target step. Users must define the desired time in which to run the second step, in relation to the first step. The user can also define a range of time in which to run the step.



SoftLinx will not run the second step until the desired or minimum time has passed.

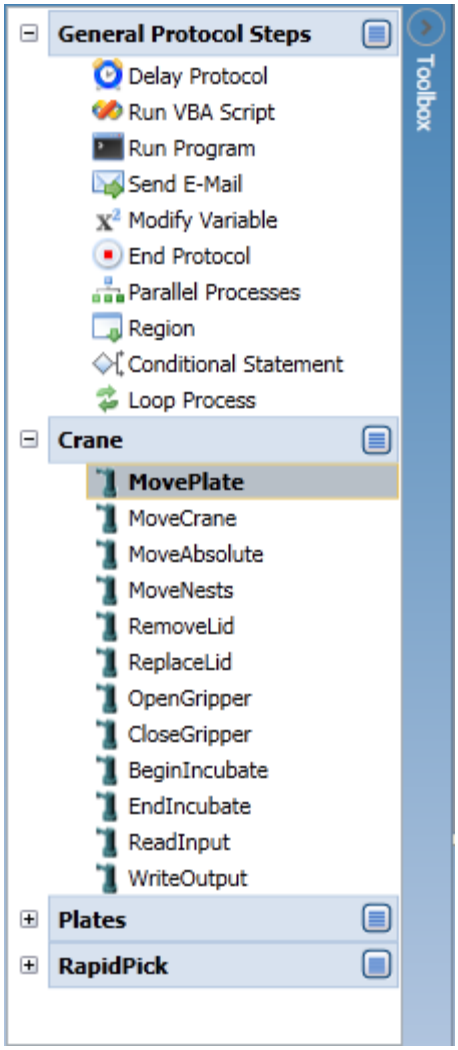
For users with SoftLinx Master installed, the maximum time slot is required for scheduling purposes.

Protocol Steps

There are several different types of steps used in protocol development. SoftLinx V has a specific set of steps the user can place in methods.

The Toolbox:

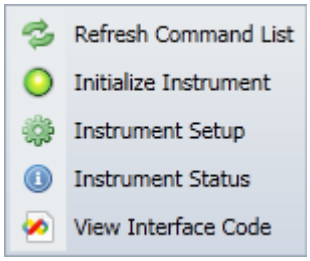
The Toolbox holds every command that can be utilized by users in a protocol. General steps are listed, as well as commands from every active instrument.



Only instruments which are checked in the Instrument List for the currently active protocol, will show their commands in the toolbox.

The toolbox is located on the left side of the protocol editor window. In addition, the toolbox can be minimized by clicking on the blue bar between the protocol tabs and the toolbox.

By clicking on the icon, or right clicking on an instrument category, the user may access additional options for that instrument, similar to those found in the Instrument Manager window:



Refresh Command List - Obtains the list of commands from the interface, and adds them to the toolbox.

Initialize Instrument - Refreshes the interface's connection to the instrument.

Instrument Setup - Opens the Setup for that interface.

Instrument Status - Opens the Status window for that interface, if available.

View Interface Code - Opens the VBA editor for the selected interface.

Protocol Steps:

Protocol steps are specific actions which will be executed in protocols. Protocol steps come in two types: Action steps and Composite steps.

Action steps perform various tasks within the protocol. In SoftLinx, the action steps include Timed Delays, Running Programs from Command Prompt, Running VBA Scripts, Sending E-mails, Modifying Variables, and Terminating the Protocol.

Composite steps house a set of action steps within itself, and will perform those sequences of steps in a specified manner. In SoftLinx, composite steps include Loop Process, Conditional Process, Parallel Processes, and Regions. The composite steps do not execute their own code, but only execute the actions within them in a defined manner.

All steps have an activate and an expand button. The Activate button can be toggled to activate or deactivate steps, and will define whether or not a step will run. The expand button can be clicked to show additional details for that specific step. For composite steps, the expand button can be used to show all of the child steps within it, or compress the step.

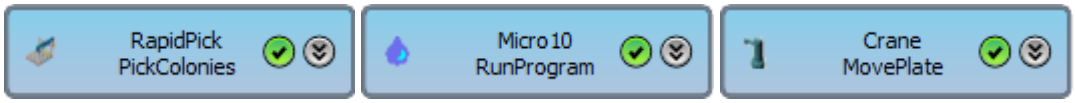
In addition, the labels of composite steps can be edited by placing the mouse over the label and clicking it.

To utilize the steps, users can simply drag and drop items from the toolbox onto the protocol. Users can drag steps onto visible arrows between steps. If a step can be placed, a drop target will be shown.

Upon dropping the step, the user will be presented with options to modify that step, if applicable. Users may also double click the step on the protocol, or right click it to access those options.

Each protocol step type has its own options associated with it. See each step's help entry for details.

Instrument Steps



All instruments have a unique set of commands which the protocol editor will present to the user. Each command is placed in the toolbox for the user to access.

When dropped onto the protocol, the instrument step will open its own setup window, if applicable. In addition, each instrument step will be identified by a steel blue background, as well as its instrument name, its instrument command, its instrument specific icon, and when expanded, details about the command issued.

See each instrument's help file for details about each command.

Run VBA Script

The Run VBA Script step allows advanced users to create their own function which they may use in protocols. Users are only limited by the scope of VBA as to what they are able to create and utilize.

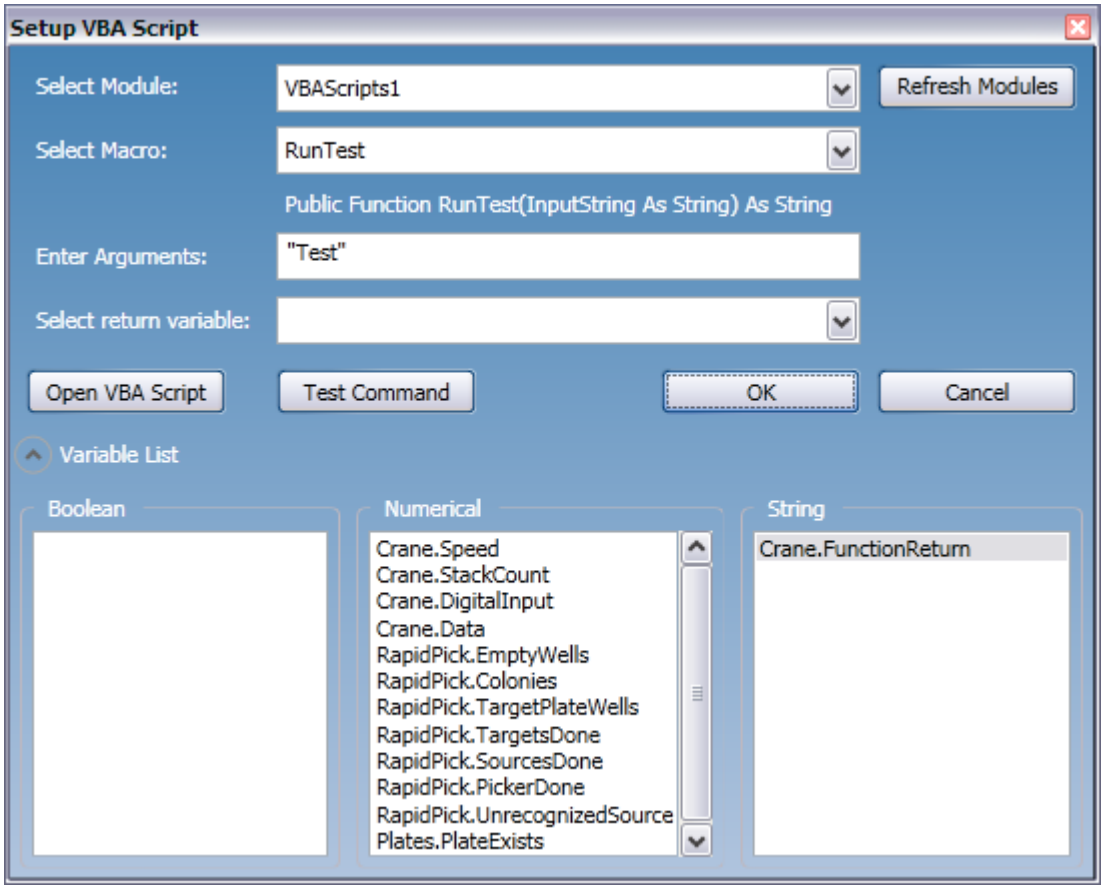
To use this step, users must open VBA and create functions. To do this:

- Click on the open VBA icon in the toolbar.

- Press Alt + F11 on the main SoftLinx Protocol Editor

- Place a VBA Code Activity on the active protocol and click "Open VBA Script."

When placing a VBA script step onto the protocol, the following window will appear:



Users must select the VBA module which the commands are defined. Users can then select the function or sub they wish to run under "Select Macro". These two combo boxes are read-only lists, which can be refreshed by clicking the "Reload Modules" button, or when the form is first opened. The list of modules is generated from all modules that are saved in the Instruments\VBA Script folder. All local users share the same module list.

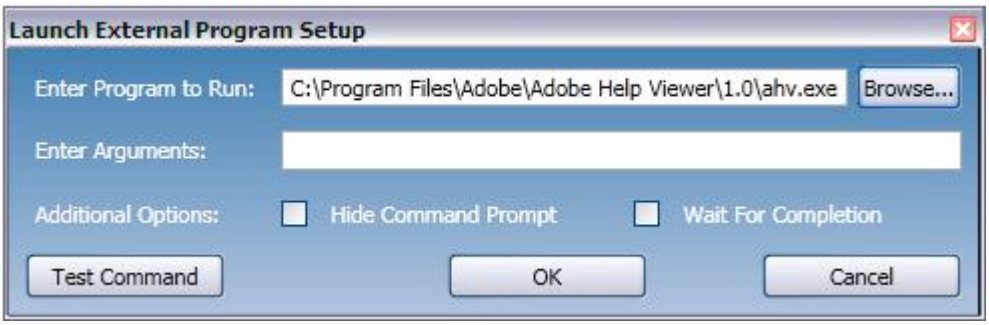
Once a command is selected, the function is shown, including its return type and all required arguments. Users may enter arguments in the adjacent text box manually, or drag and drop the appropriate pre-defined variables to satisfy requirements.

When opening the VBA Script, users have access to all installed instruments on the list, as defined on the Instrument Manager. Users can then apply commands from the instruments in their own predefined functions through VBA. Note that all instruments on the list are accessible, whether or not their interfaces are open. Users may then test their functions by clicking the "Test Command" button to ensure their functions work as intended.

Users should refer to the VBA help files for assistance in creating VBA functions.

Run Program

The Run Program step will run one command line string or open an executable with optional arguments.

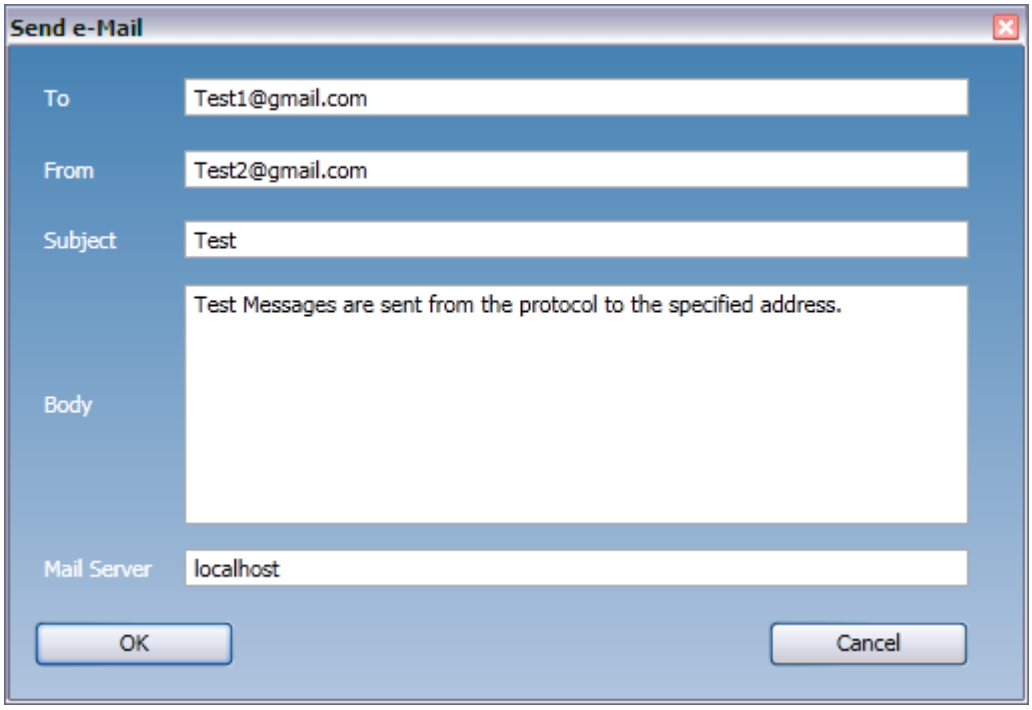


Users will enter the path of the executable and the arguments in the proper text boxes. Users may also test the command by clicking on the "Test Command" button.

The "Hide Command Prompt" option will run the command without the command prompt window showing up at runtime, if applicable. The "Wait for Completion" option will force the protocol to wait for the complete execution of the command before continuing on to the next step.

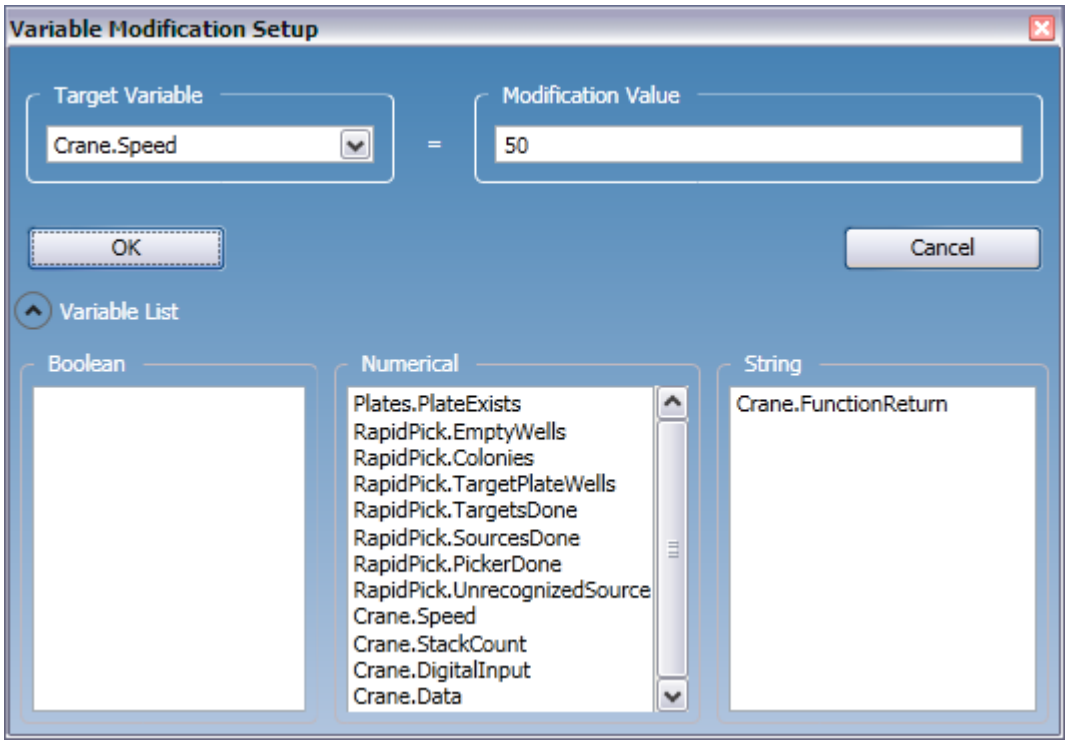
Send E-Mail

The Send E-mail Notification step will allow a user to send out an email message at a certain location within the protocol. The user simply adds the basic fields to an email message: To/From addresses, subject, and body paragraphs. Additionally, the user must provide a mail server host which will be used to send the mail.



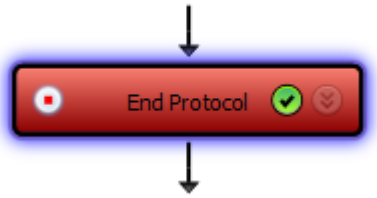
Modify Variable

The Modify Variable step will change an instrument or protocol variable to a different value. Users can input direct values or functions to define the new value of a variable.



SoftLinx will ensure that the result of the modification value is valid before the user can confirm the step. Invalid entries will not be accepted.

End Protocol

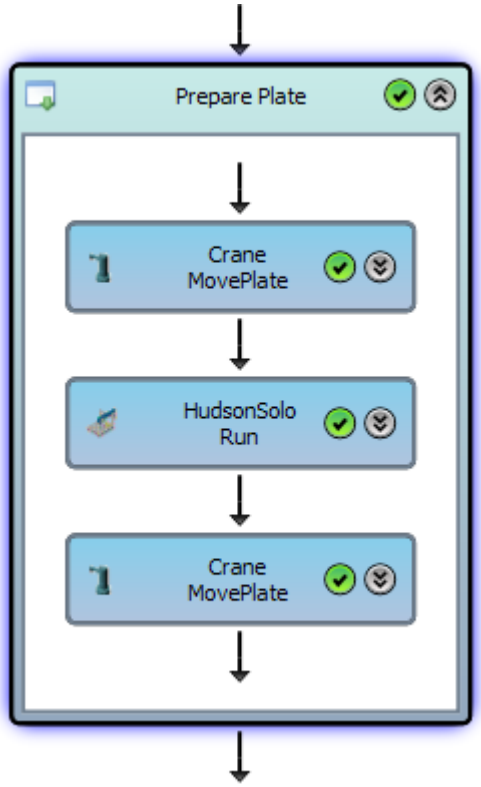


Simply stops execution of the protocol. This can be useful in conditional statements, or used as stop points if a user wants to only run a portion of a protocol without making major modifications to an existing protocol.

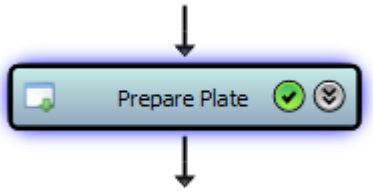
Region

Regions are collapsible actions which can hold any number of protocol steps. Users may place steps within a region, label the region, condense the region, and even deactivate it so that no steps within the region will execute at runtime. Region steps are used for organizational purposes, and do not have any special execution rules associated with them.

Expanded:



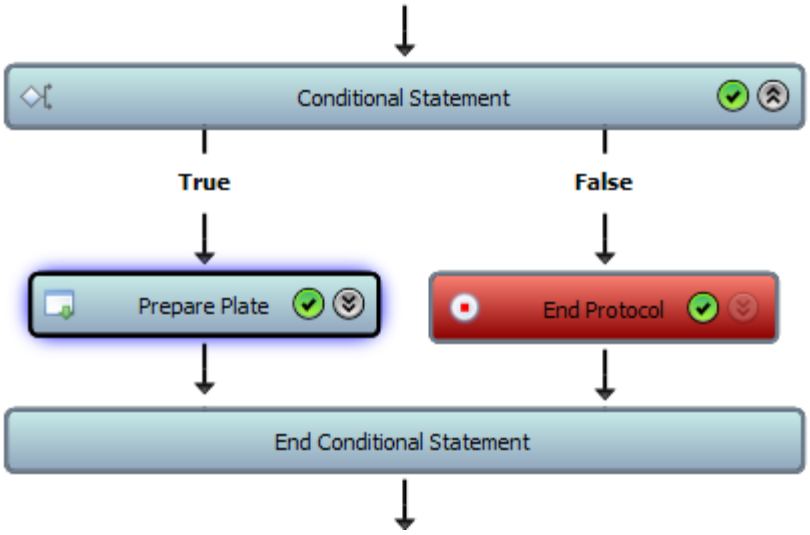
Condensed:



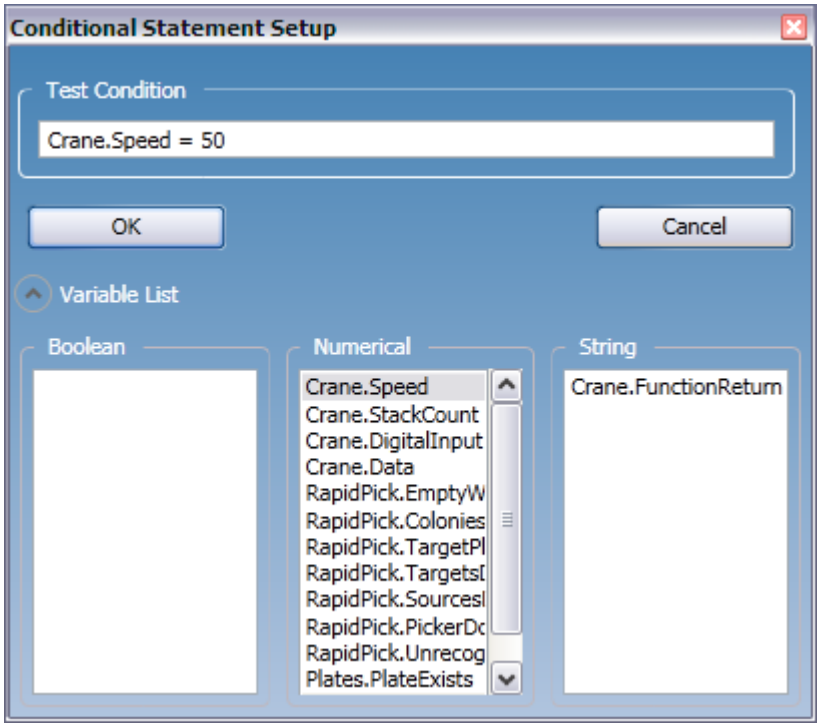
Regions do not have special execution rules, and will run the steps in order, as if they were never placed within the region.

Conditional Statement

Conditional statements are used when a decision to run only one set of specific steps is necessary.



Based on the result of the current condition defined, one and only one of the two branches will execute its set of steps. Users can define a condition using the setup screen below:

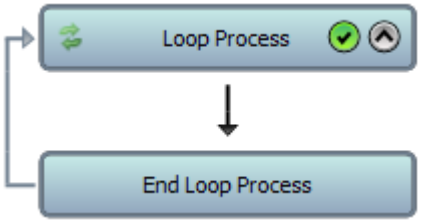


Users must write a function or statement which must evaluate to a "True" or "False" response, such as the example above, Crane.Speed = 50. If the conditional statement the user selects turns out to be true (If Crane.Speed is 50, then the statement is true), then the left branch will execute its steps. If the statement is false (If Crane.Speed is 40, then the statement is false), the right branch will execute.

Note that any protocol-defined or valid interface variable can be used in the statement. SoftLinx will detect if the function or statement is able to evaluate to a True or False response.

Loop Process

Loops are composite steps, which will execute a set of activities within itself repeatedly for a set number of iterations. Steps must be placed between the loop and end loop boxes to be considered as part of the loop, and repeat as defined.



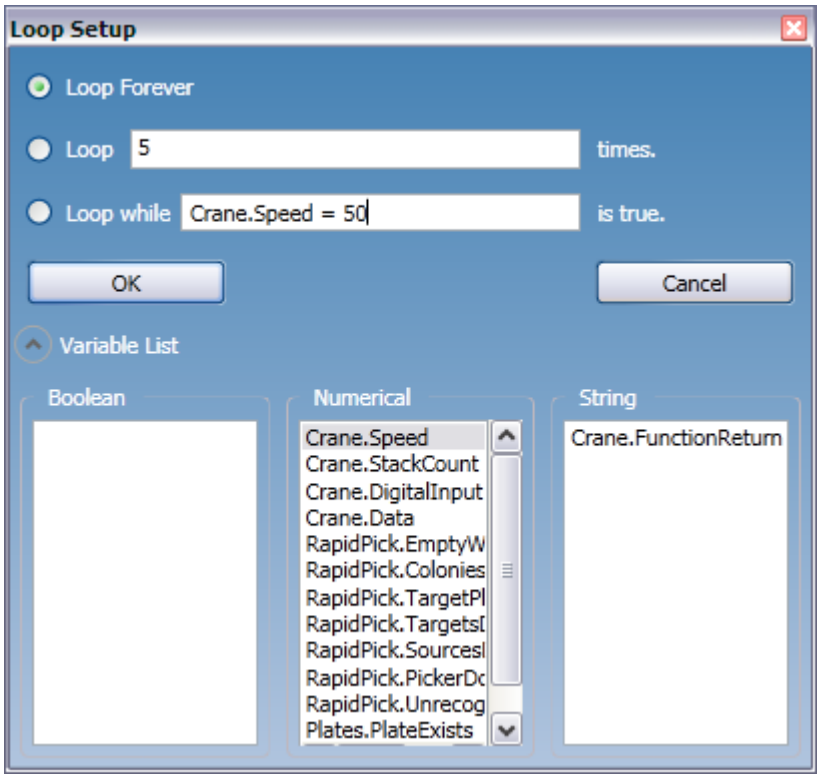
Users can define the number of times a loop runs by one of three ways:

- Forever (until stopped by an internal step or the stop button).

- Finite number, either predefined or variables.

- Loop by condition.

Loops are defined using the setup window shown below:



Finite loops are predefined, and will always run the loops for that number of iterations unless stopped by the stop button or an internal step. Loops that run forever can only be stopped by an internal step or the stop button.

Conditional loops will run depending on whether or not a condition is satisfied. Users define a conditional statement, which if evaluated to true, will allow the loop will run. As long as that condition is true at the

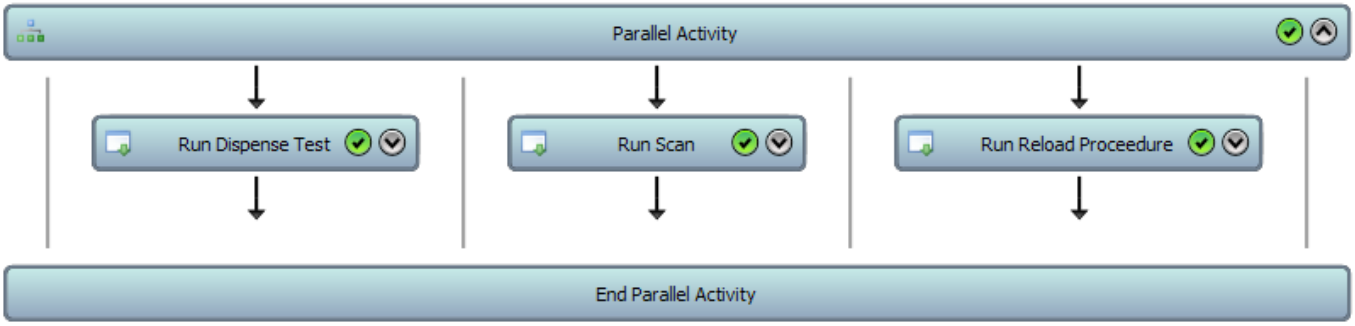
beginning of the loop's execution, the loop will continue to repeat itself. If the condition becomes false by the time the loop executes, or repeats execution, the loop will no longer execute, and the protocol will continue on to the next step following the End Loop box.

Conditions are checked before a loop executes its process, and will re-evaluate the condition before repeating the process each time to determine if the loop should run.

Note that any protocol-defined or valid interface variable can be used in the conditional statement for the loop. Invalid variables will not be accepted.

Parallel process

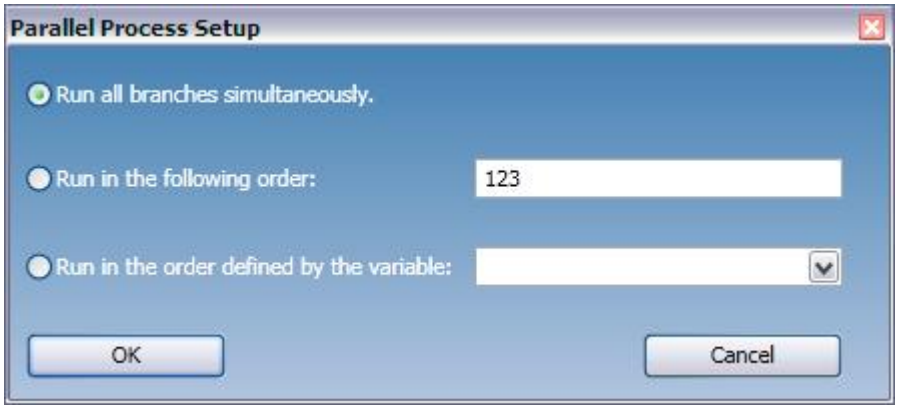
Parallel processes can run multiple sets of steps, either simultaneously or in a predefined order.



Each branch holds its own set of steps. When this process begins, it will begin execution of each branch, either simultaneously or in a specified order. This process will not complete until all branches have been executed, and all child steps have been completed.

Users may add branches by dragging steps onto the silver lines shown between steps. Only one silver line is shown if no steps exist within the parallel step. Users may then add steps to branches by placing items on the arrows.

Users can define the way this step will execute its branches through the following window:



By default, all branches will run simultaneously, and only one time.

When selecting "Run in the following order", the user can type in a series of characters that denote an order in which to run branches or to run multiple branches multiple times. For example, if a user selects this option and the input string is 123, the parallel process will run branch 1, then branch 2, then branch 3, sequentially. Alternatively, if the user enters the string 321, the parallel process will run branch 3, then branch 2, then branch 1, sequentially. To run branches simultaneously, use (). If a user enters the string (12), then the parallel process will run branches 1 and 2 simultaneously, and not run branch 3 at all. This can lead to more complicated strings and execution of branches. Advanced Example: If a user enters the string (12)3(12)11(123), then the parallel process will run branches 1 and 2 simultaneously. Once the set is done, it will run branch 3 by itself. Then, it will run branches 1 and 2 simultaneously. Then, it will run branch 1 by itself, twice, in sequence. Finally, it will run all 3 branches simultaneously before it is completed.

Should a user have more than 9 branches, the user must use A-Z to denote branches 10-35. Branches 36 and above cannot be run using this method, however it is very uncommon to have 36 or more branches under one sequential protocol step.

In addition, users can also select a protocol string variable to define the string used to control the execution of branches. The string must match the same format as specified above. Invalid characters will be ignored.

Protocol Tutorial

Protocols are sets of instructions given to the instruments to perform in a specified manner.

The following instructions will advise on how to create a simple protocol. Please note that any protocol can be created in multiple ways.

1. Define your task.

It is very important to clearly understand the goal of the protocol before creating it.

Problem: We wish to fill 20 micro plates with solution, using the Micro 10 (liquid dispenser) and the Plate Crane (micro plate handler).

A sample outline to solve the problem can be seen below:

Take a plate from a stack on the Crane, and place it in the Micro 10.

Run Dispense Program 3 on the Micro 10.

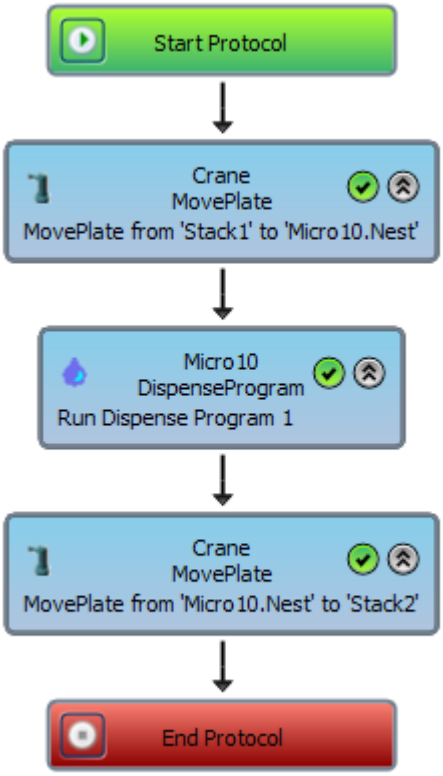
Move the plate from the Micro 10 to a different stack on the Crane.

Repeat the process 20 times.

The following example is a short protocol, and can be completed in about 4 steps.

2. Create the protocol in a linear fashion.

It would be best to create a protocol for one plate, based on the outline above. Split the protocol into steps.



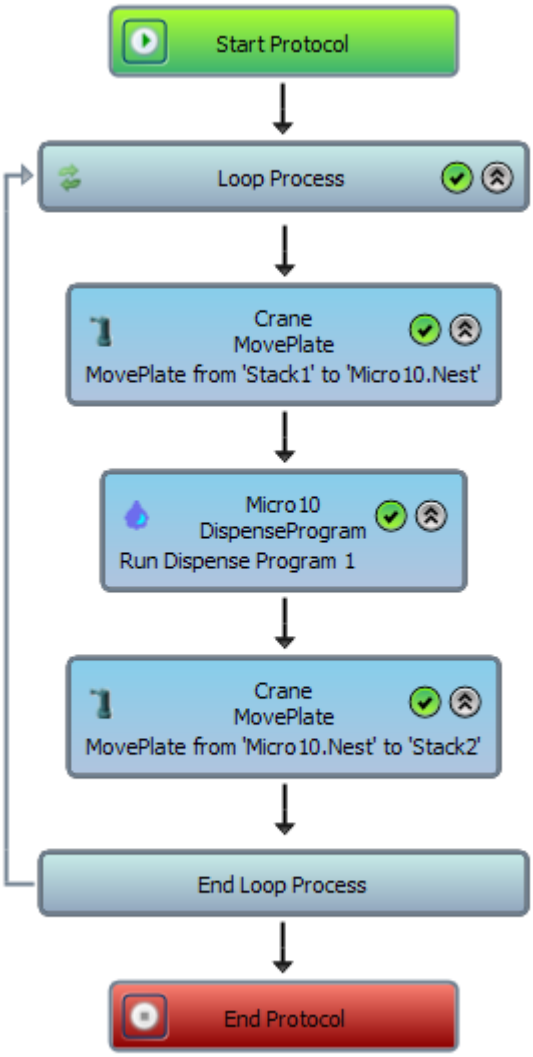
All steps can be dropped from the toolbox to the canvas to create the pattern above. Protocols run from start to finish, top to bottom.

3. Test the protocol.

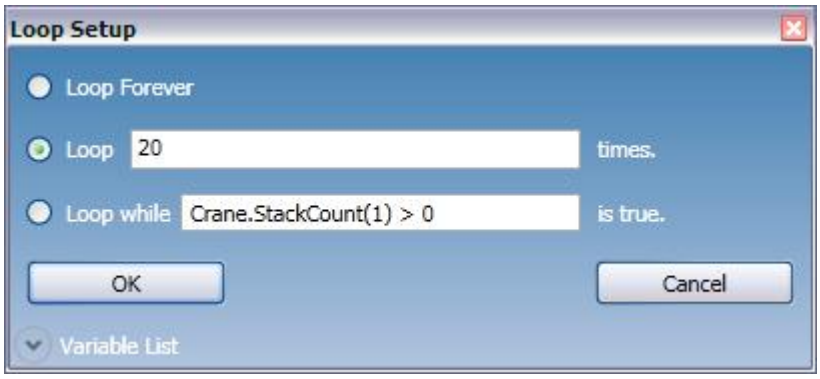
This should complete the goal for a single plate. It is recommended to run this protocol now, to ensure that it acts the way you want it to run, and that all instruments are running properly.

Ensure that the correct plate types and initial plate locations are defined for this protocol. A 96 well plate, with initial plate location at stack 1, is recommended.

However, we are not done. This protocol is supposed to run 20 times. To repeat parts of a protocol, use a loop step, and place the items you wish to repeat within the loop.



There are multiple ways to set up this loop. Users can either set the loop up to run forever and empty stack 1, or run the loop a specific amount of times, or use variables to determine how many times a loop should run. In the example below, we can use the instrument variable "Crane.StackCount(1) > 0" to tell the loop to run until the number of plates in the Crane's stack 1, as detected by the Crane, is 0.


Loop Setup
Loop Forever
Loop
20
times.
Loop while
Crane.StackCount(1) > 0
is true.
OK
Cancel
Variable List

Protocol Tutorial - Advanced

Advanced protocols require multiple steps of various types to complete. A sample advanced protocol which utilizes the Rapid Pick is shown below.

1. Define your task.

Situation: We have a bunch of bacteria colonies in agar plates that we would like to pick, using the Rapid Pick colony picker, and place into 96 well micro plates. The micro plates must be pre-filled with solution from the Micro 10, and later sealed by a plate sealer of some type. We also have a plate crane to move plates between all of the instruments.

It is highly recommended to break up this task into parts. With SoftLinx V, the user can use region activities to label each part of the protocol separately.

This protocol has 3 parts.

A. Initial Loading of the colony picker. This portion of the protocol will only be done once.

B. Actual Colony Pick. The picker will be reloaded every time plates are needed.

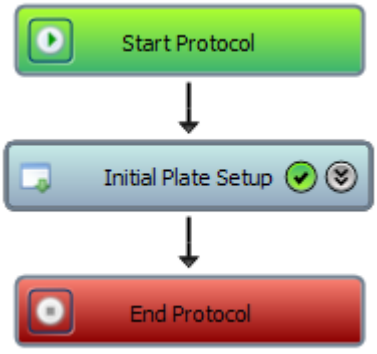
C. Final unloading of the picker. This is performed once, at the end of the protocol.

For this example, the protocol requires the following interfaces:

Crane - Micro Plate Handler
Micro10 - Liquid Dispenser
RapidPick - Colony Picker
Brandel - Plate Sealer

2A. Build the Initial Plate Setup region.

We can focus on one part at a time. First, we can focus on the initial loading of the colony picker. Place a region activity on the canvas, and label it "Initial Plate Setup".



Now, we must define what we need to do to set up the Rapid Pick for picking. Generally, preparation should go like this:

- Move a lidded colony source plate from a stack to the RapidPick.

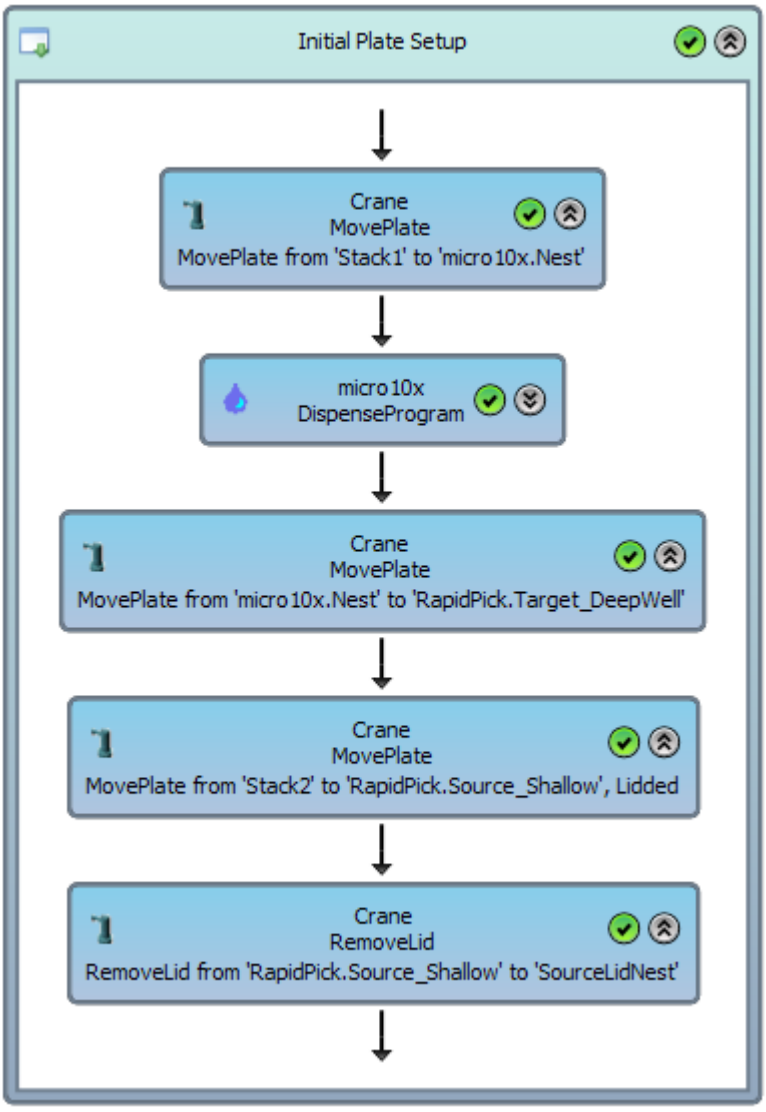
- Remove the lid from the source plate.

- Move a new target plate from a stack to the micro 10.

- Dispense liquid to the target plate.

- Move the prepared target plate to the RapidPick.

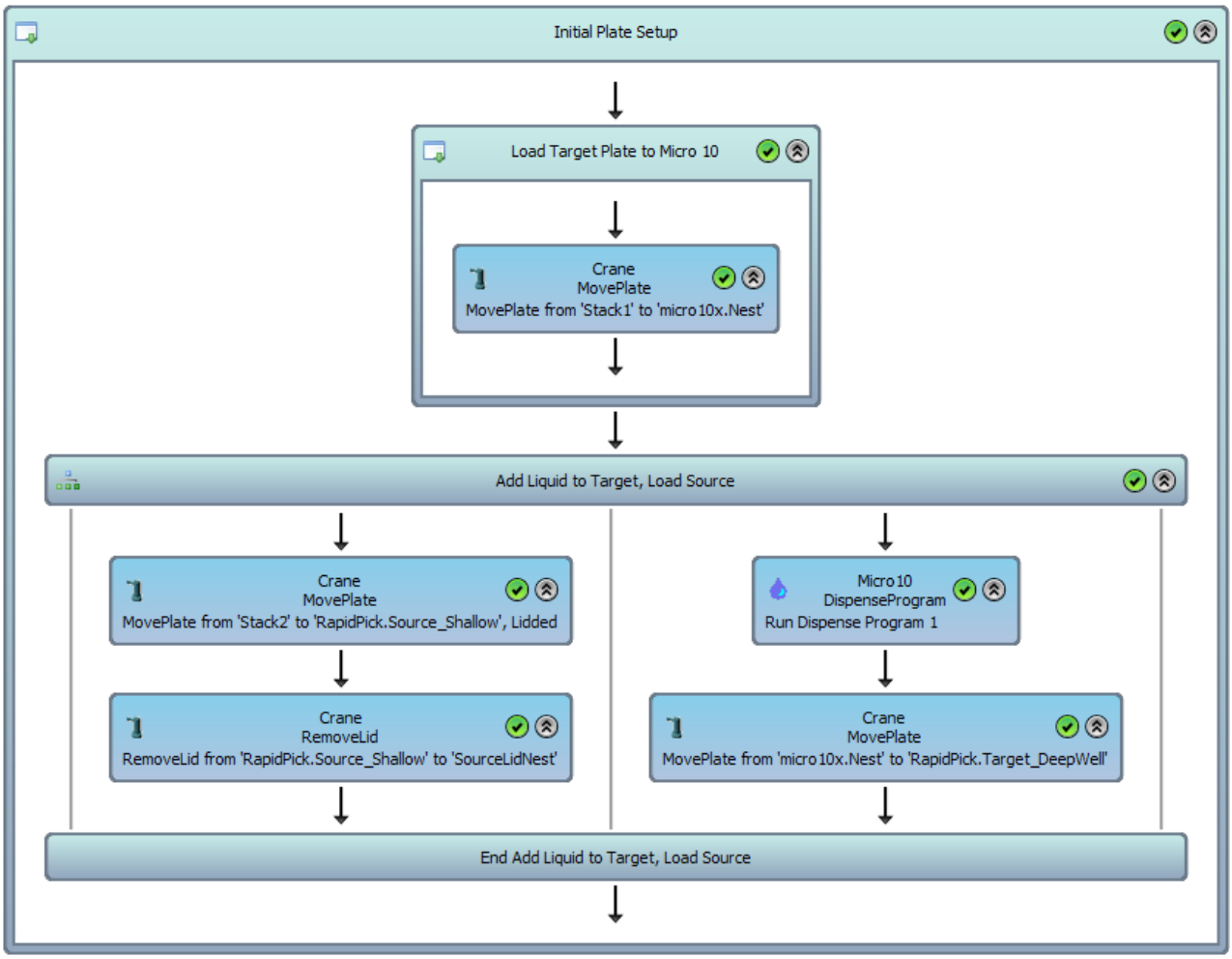
We can start by placing all the steps in a linear fashion, like so:



This will work and load the plates. However, we don't have to wait for the micro 10 to be finished with the target plate before we move the other source plate. We can modify this part of the protocol even further by using a parallel process activity to run multiple instruments at once.

Place a parallel process onto the canvas. Place the Micro 10 dispense step and the move target plate step in one branch, and the two steps that handle the source plate in another branch.

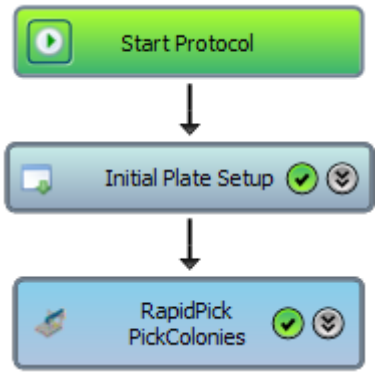
This part of the protocol will look like this:



This allows the Crane to prepare the source plate while running the Micro 10 on the destination plate simultaneously, saving time to run the protocol.

2B. Build the Pick Colony region.

The next part of the protocol will consist of picking the plates, and reloading the picker with new plates. Place this after the initial plate setup region. Note that you can also minimize regions.



After a pick step is complete, we will need to do some type of clean up. The RapidPick will need to either

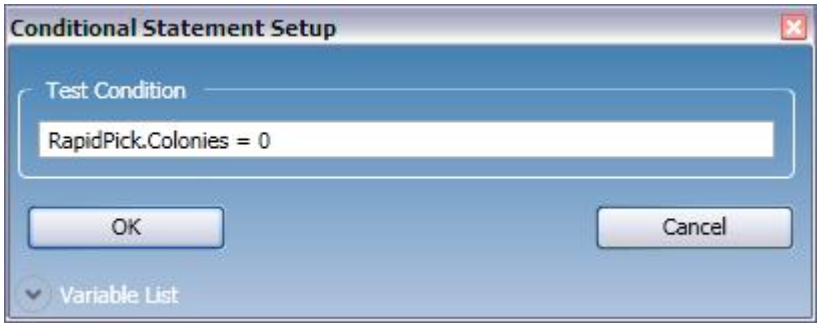
remove the source plate, target plate, or both, before starting a new pick.

In this case, instrument variables can be used to make decisions within the protocol. The Rapid Pick has two variables which can be checked to see the status of the plates being worked on.

RapidPick.Colonies - Denotes the number of colonies remaining on a source plate. If this value is zero, remove the plate.

RapidPick.EmptyWells - Denotes the number of spaces left in a target plate that are yet to be filled.

When a decision needs to be made, the use of a conditional statement is generally required. Place a new conditional statement on the canvas. Enter the following condition:



When this statement shown is set to true, then the conditional statement's true branch will activate. Replacing the source plate will consist of the following steps:

- Replace the lid of the used source plate.

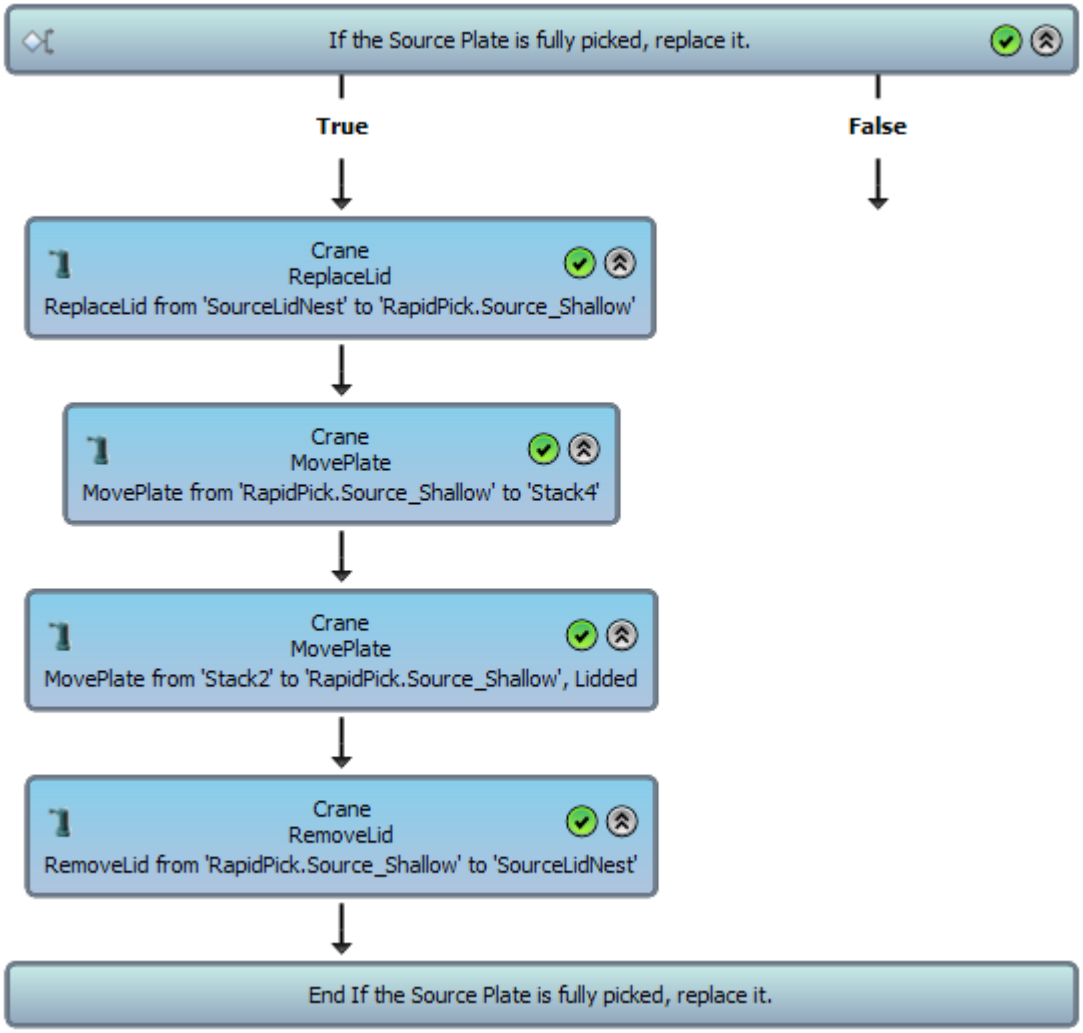
- Move the used source plate to a "used source stack".

- Move a new source plate to the source location on the RapidPick.

- Remove the lid from the new source plate.

Place the replace lid step, move plate steps, and remove lid steps into the true branch of the conditional statement. Do not place anything in the false branch.

The resulting conditional statement should resemble something like this:



This will replace the source plate when necessary.

The target plates are handled in a similar manner, although additional steps must be taken when preparing new target plates and storing finished target plates. The steps are as follows:

- Check the target plate. If it is not ready to be removed, do nothing.

- If the plate is ready to be removed, move the finished target plate to the plate sealer.

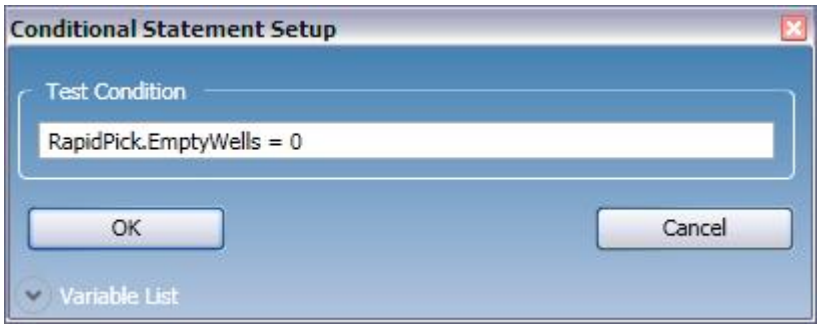
- Begin sealing the plate. While sealing the plate, move a new target plate to the micro 10, if there are still source plates that can fill the remaining target plates.

- Remove the finished target plate from the plate sealer, and place in a finished targets stack.

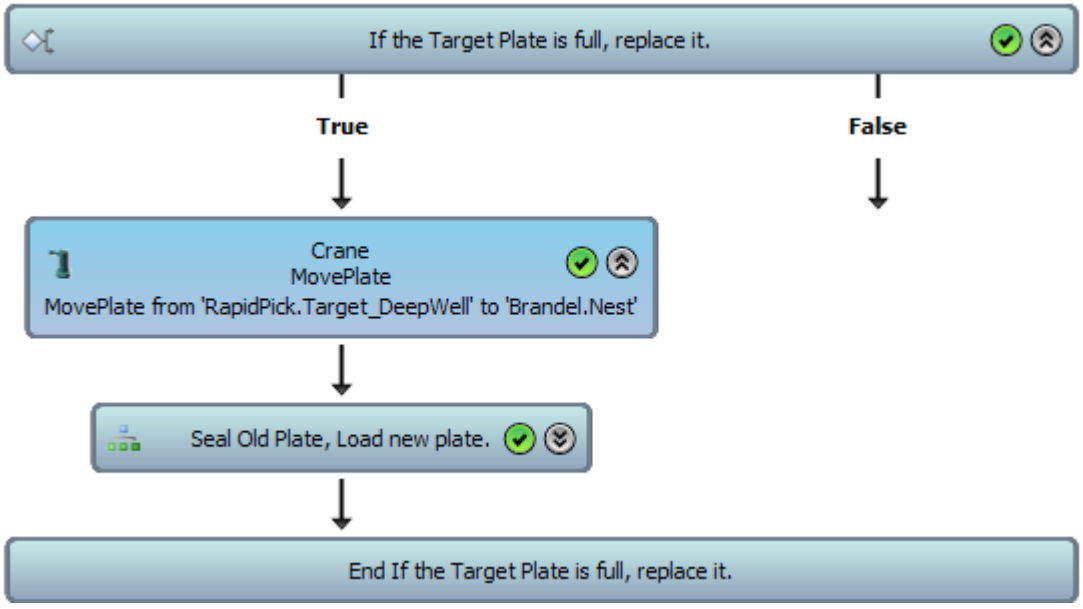
- Remove the newly filled target plate from the Micro 10, and move it to the target nest of the Rapid Pick.

As these steps have a few decisions to be made, place steps down, one at a time. Start with a conditional

statement that checks the RapidPick.EmptyWells variable to see if we even need to replace the target plate.



Place a move plate step, directed at the plate sealer, and a parallel process activity labeled "Seal old plate, load new plate." as shown below.



Next, expand the "Seal Old Plate..." process. We need to run simultaneous actions here. We are going to simultaneously seal the old plate, while preparing the new plate, if a source plate is still available with colonies to be picked. The steps of sealing the old plate are as follows:

- Run the sealing program on the plate just placed there.

- Move the plate to a final destination.

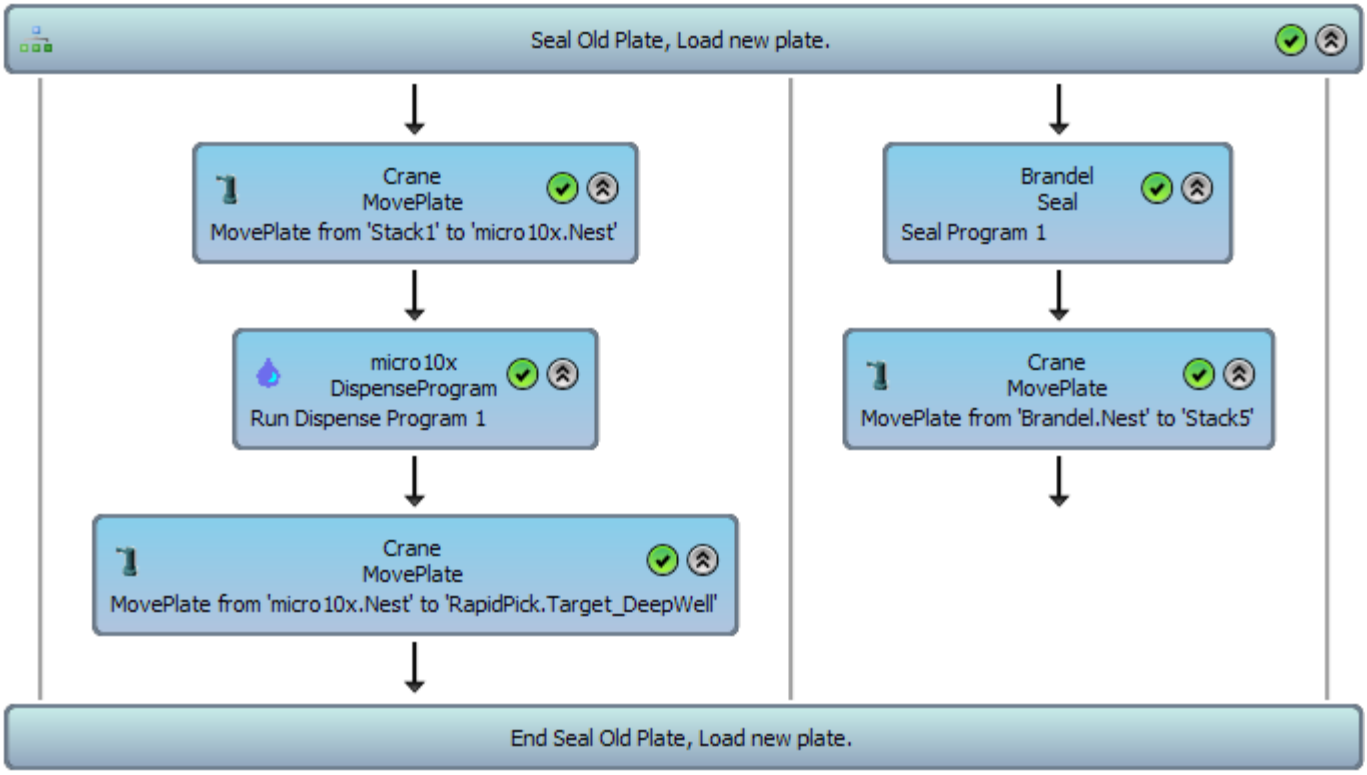
The steps of preparing the next plate should be the same as in the initial plate setup region:

- Move a new target plate to the Micro 10.

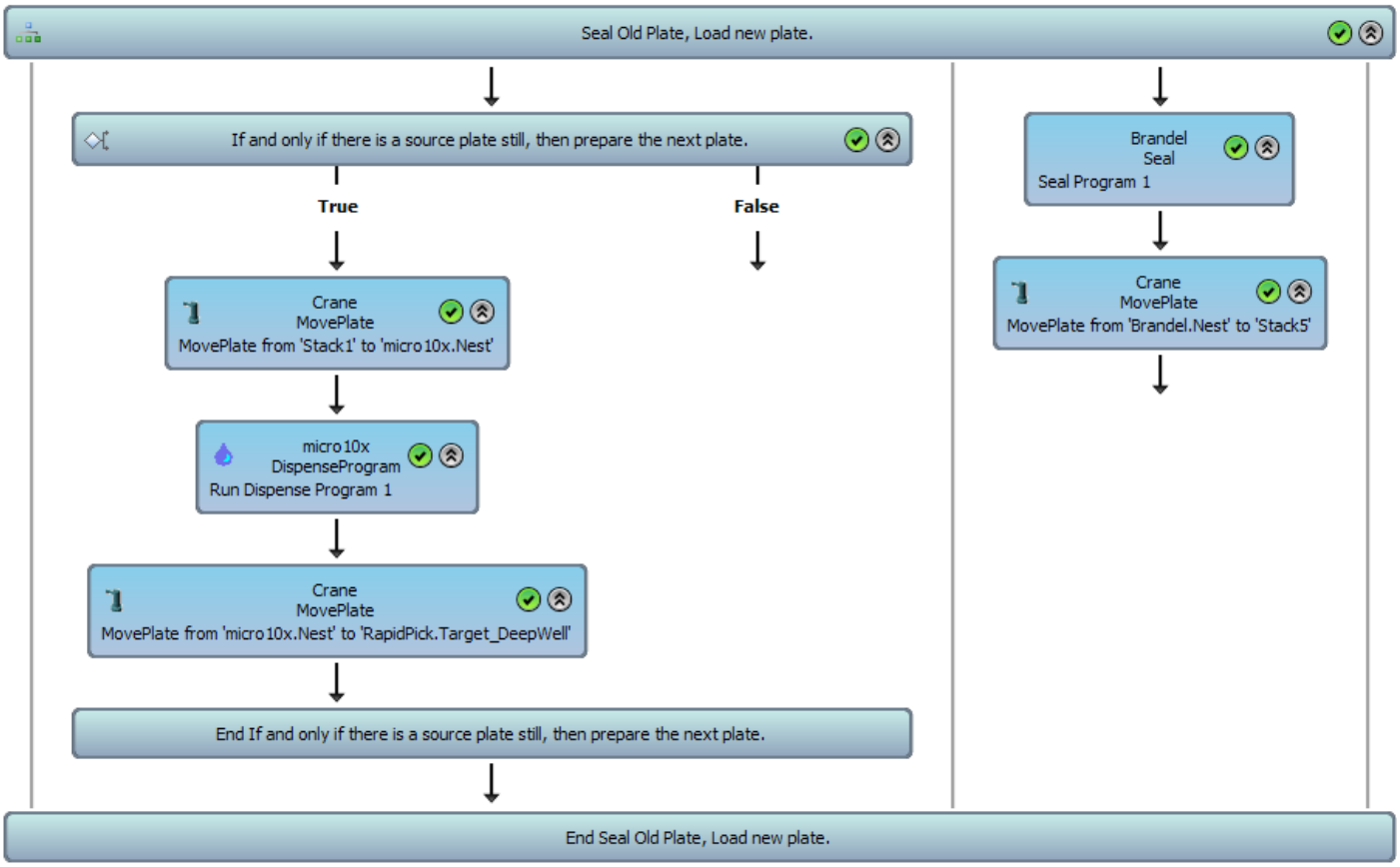
- Run a dispensing program on the new target plate.

- Move the new target plate to the Rapid Pick.

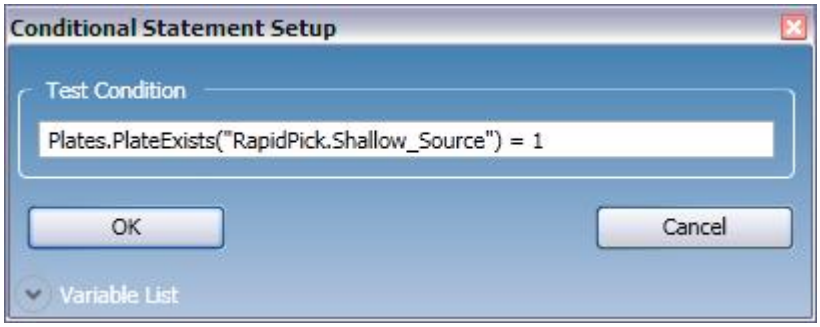
As these are two separate processes, we can split them in a parallel process, as shown here:



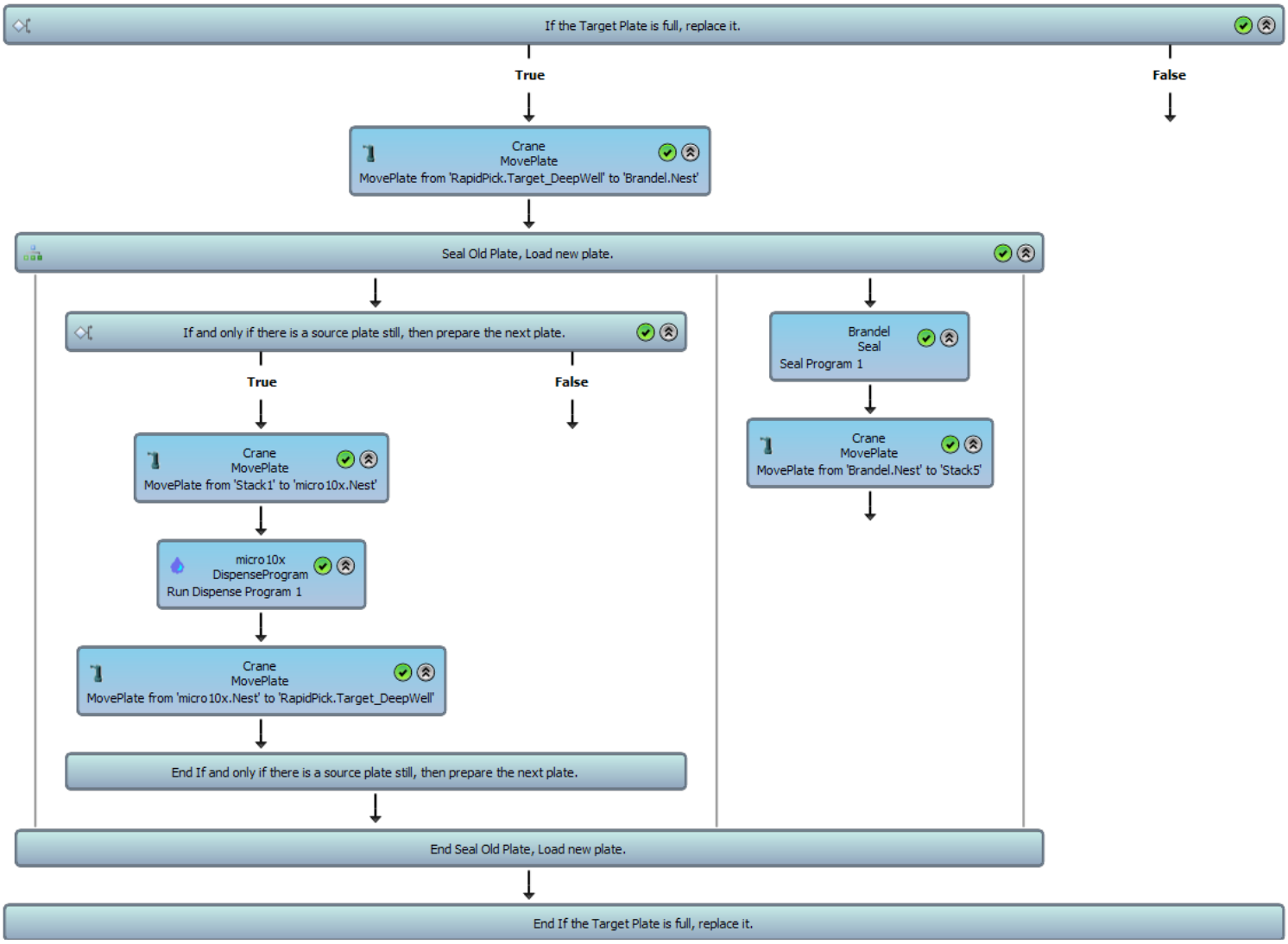
However, we only want to prepare the next plate, if a source plate is still present on the Rapid Pick. If the source plate has been removed, and not replaced from the previous conditional statement, we will not prepare a new plate. Since this is a decision, we need to place a conditional statement in the new plate process branch, like this:



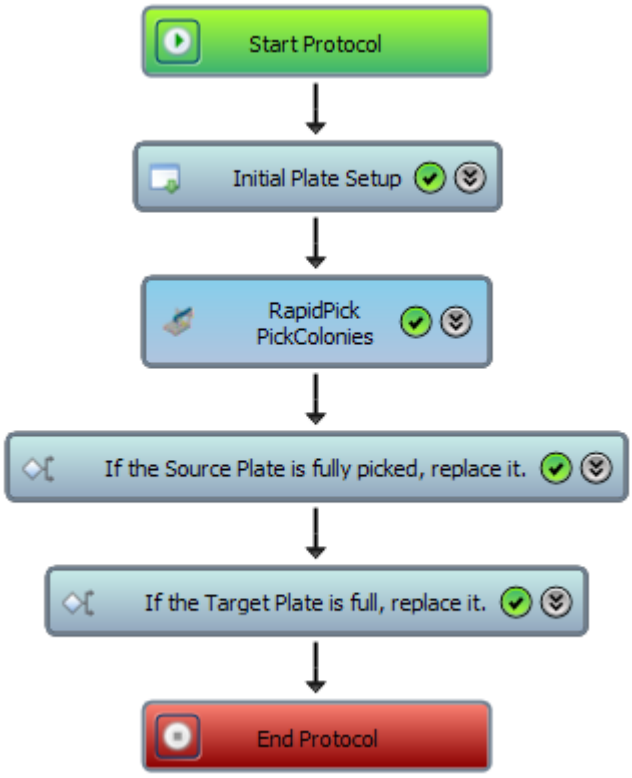
To detect a plate at a specific location, we use the statement Plates.PlateExists("X"), where X is the name of the nest where the plate is located. By typing the condition below, it will only prepare a new plate if a source plate is ready to go on the Rapid Pick source nest.



The full "Replace Target Plate" conditional statement, expanded, should look like this:



So, for a protocol, we currently have the following:



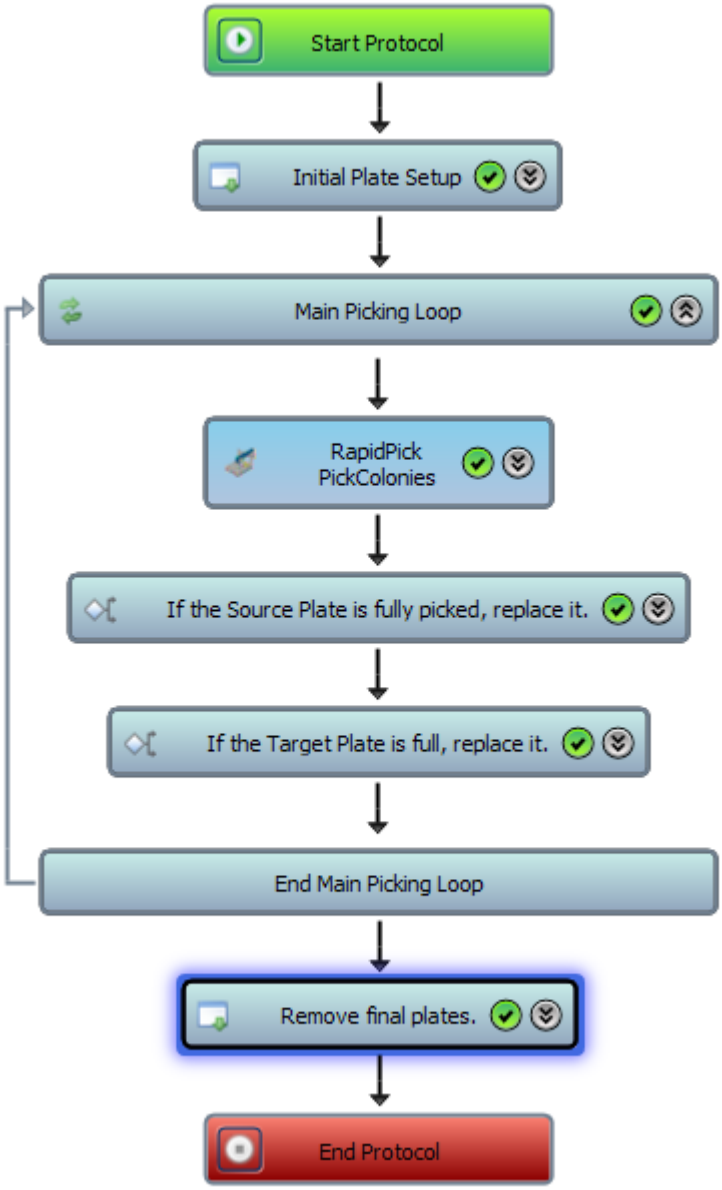
This may work for one plate. However, we want the protocol to repeat the picking and replace plate steps multiple times. We can place the last three items in a loop. We only want to run the pick step if there are plates in the source and target nests of the RapidPick.

To check that in the loop, enter the following statement in the "Loop While..." textbox:

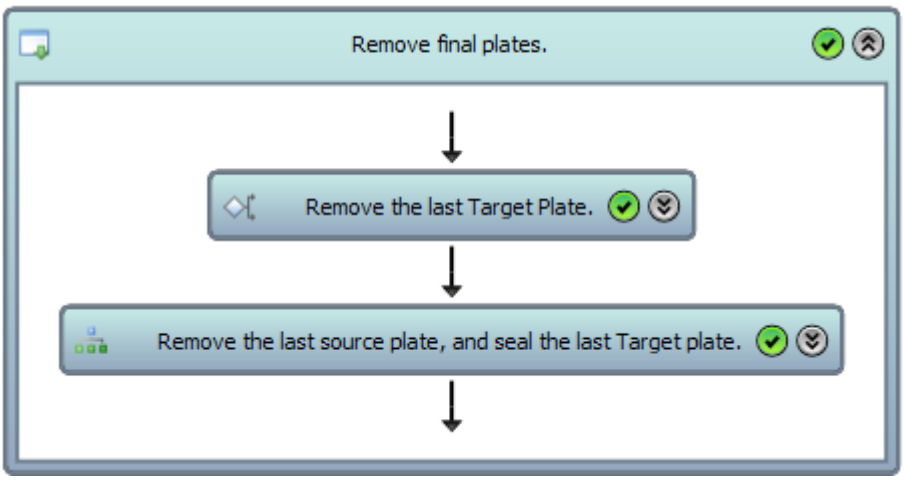
(Plates.PlateExists("RapidPick.Target_DeepWell") = 1) And
(Plates.PlateExists("RapidPick.Source_Shallow")=1)

If there are not, then we know that we are either out of source plates or out of target plates. At this point, we must end the loop and initiate a cleanup routine, as we know we are at the end of the protocol.

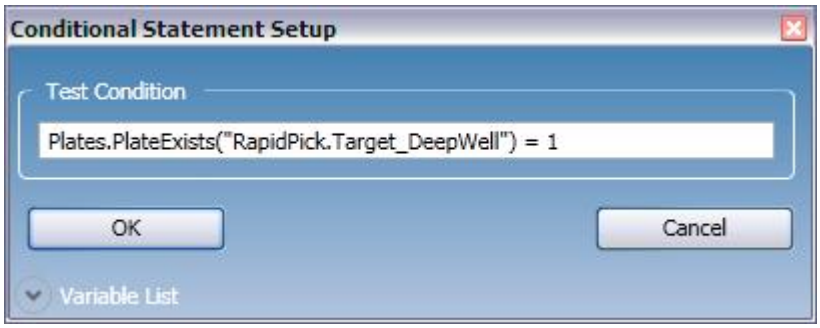
2c. Final Cleanup



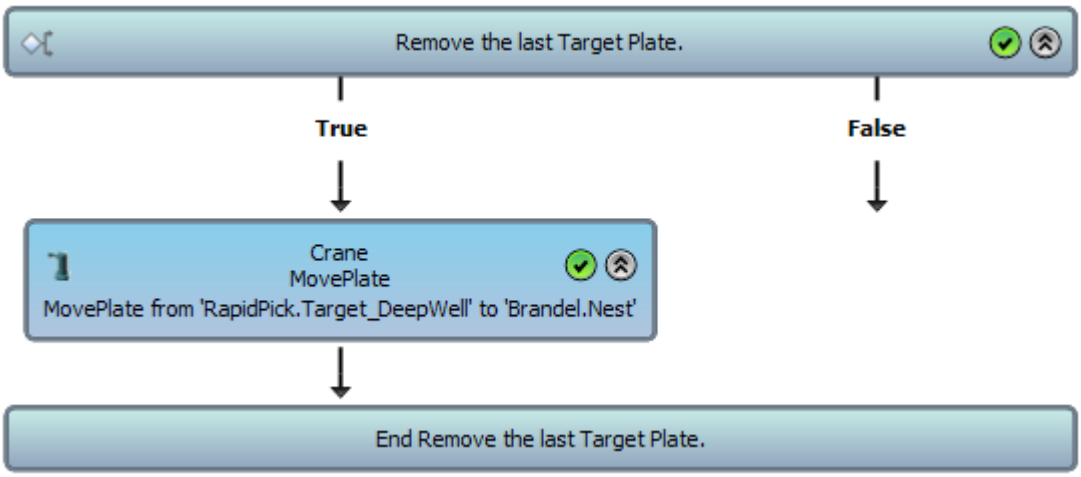
The last step, "Remove final plates", is a region activity with processes on removing the last target plate, and the last source plate. Place a region activity after the colony pick loop.



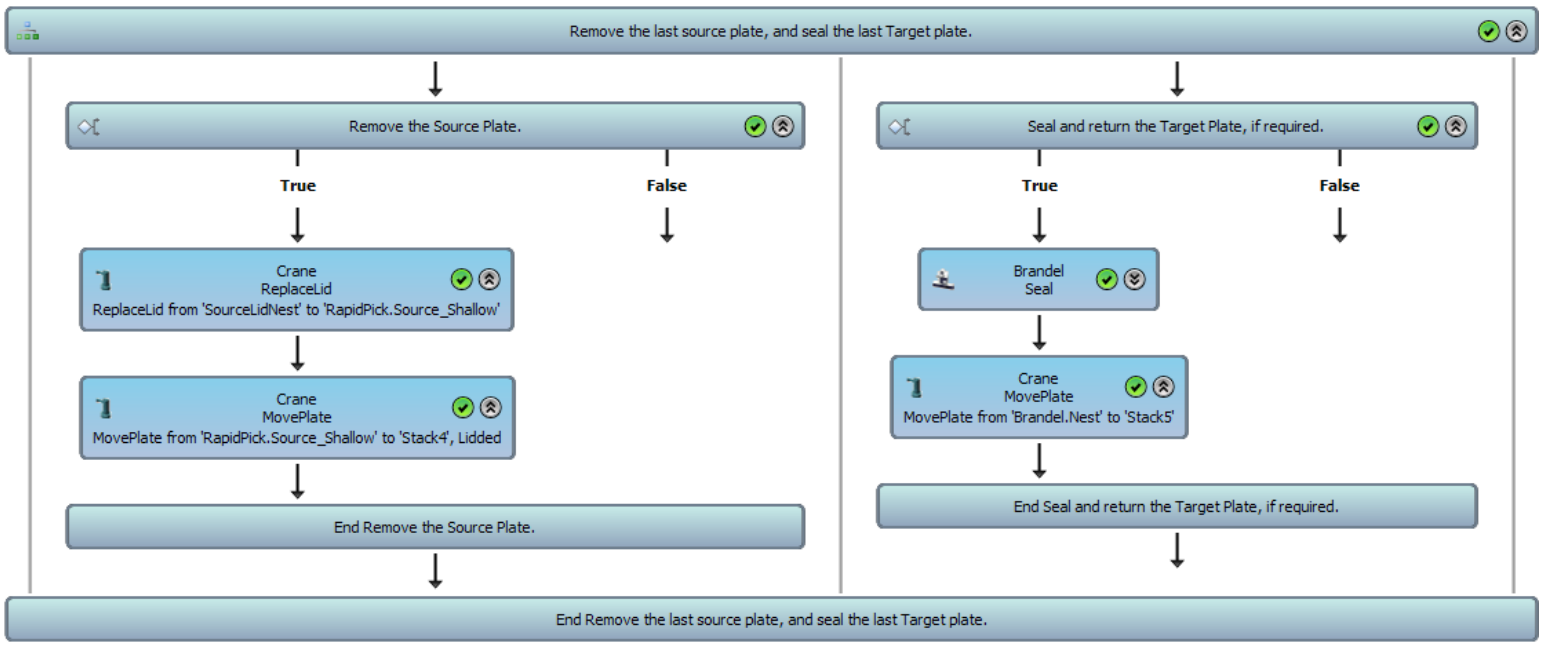
We will then check to see if there is a target plate that needs to be removed. Use a conditional statement, which tests the statement below:



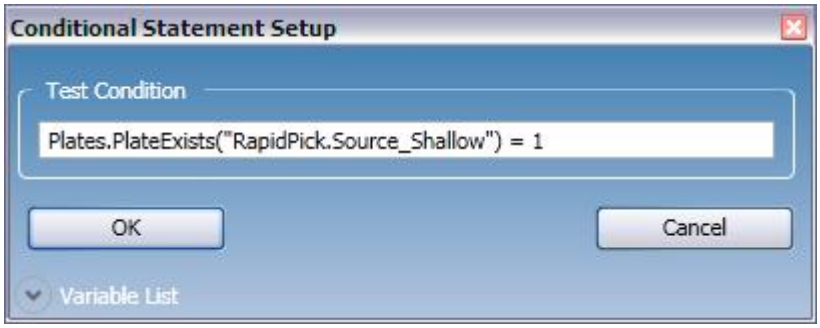
Then, place in the true branch, a move plate step from the target nest of the RapidPick to the plate sealer.



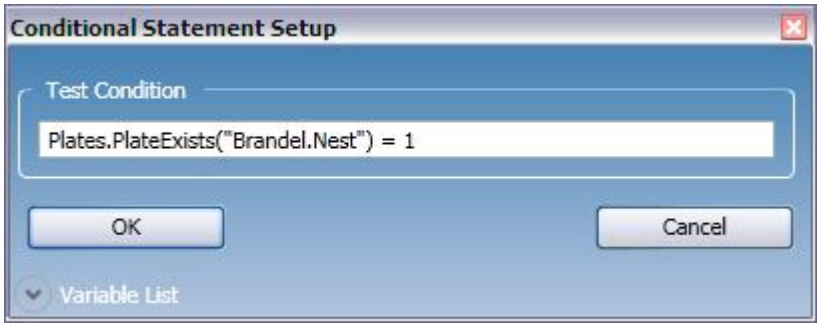
Finally, we can simultaneously check to see if there is a plate to seal, and check to see if the source plate has to be removed. Simultaneous actions require the use of the parallel process step. Using a conditional statement in each branch, we can check for a source plate to remove and a target plate to seal.



Like the previous steps beforehand, check for a source plate using the following:



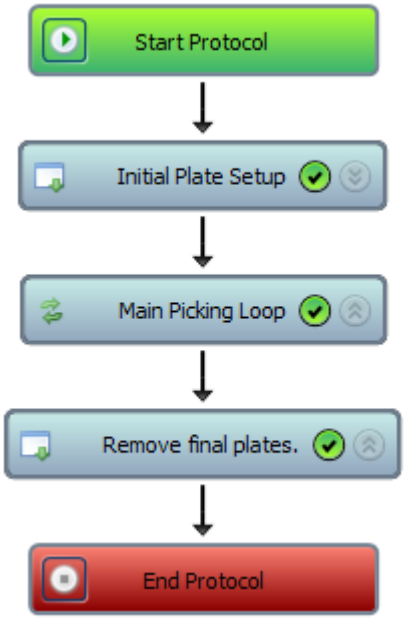
Check for a target plate in the sealer using the following:



3. Test the protocol.

Finally, once this part is done, **test** the protocol using empty plates and no liquid. Ensure that all of your source plates and destination plates are defined, and all initial plate locations match those used by this protocol. By testing the protocol, you will ensure that the protocol is written properly and no media is wasted on a faulty run.

The final protocol can condense to this:



This concludes the protocol tutorials.