HCA Tech Note 110

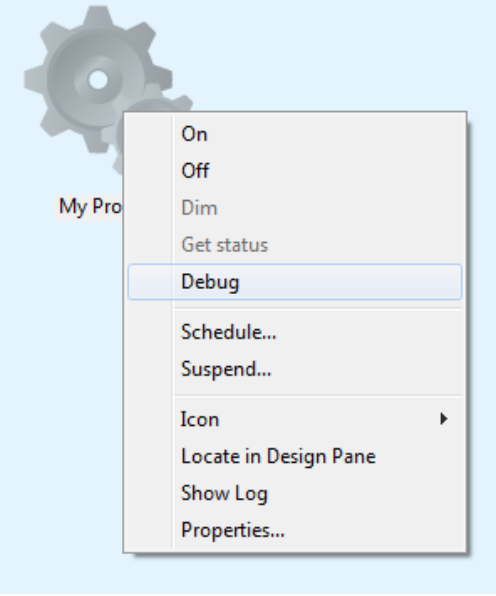
The HCA Visual Program Debugger

The Visual Program debugger, a feature only in HCA Plus, can help you find problems in programs that you create. Like a development environment for more traditional programming languages, using the debugger you can stop execution of the program at various points, examine and modify state, and control the flow of execution.

Don't think of using the debugger only to find and fix problems. Programs can be complex if they must handle a variant of conditions, some of which might not occur often. Not all the sections of a program may get executed in the normal course of events. Using the debugger, you can execute those "little used" program paths so you know that the program is fully functional before deploying it into your home environment.

This technical note covers in brief detail the top 10 debugger features that you should know about. More detail is contained in the HCA Used Guide.

1: Starting a program in the debugger



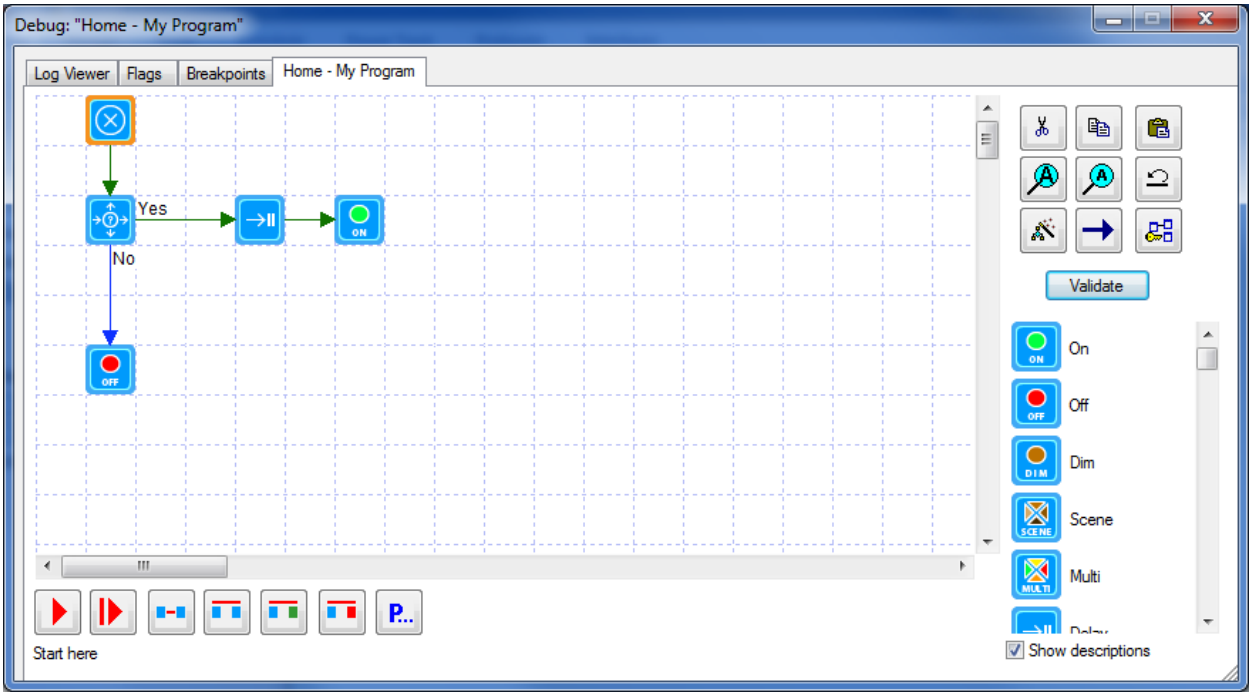
The simplest way to start a program is to right-click on it and select debug from the popup menu.

The program opens in the debugger which looks a lot like the Visual programmer with additional buttons in the debugger control area below the programming canvas.

The key fact here is that the program has not been started yet. When in the debugger, the current element – the element that is about to be executed – is outlined in orange. Since the program has not yet started, the Begin-Here element is the one outlined in orange.

The next thing you might notice is that the debugger opens in its own window separate from the main HCA window. You can minimize the debugger window to the task bar if you want. Since it is a separate Window you can continue to use all the features in HCA during a debugging session, for example controlling devices, changing schedules, using the interface tools, etc.

To close the debugger, use the "close button" – the "X" – in the upper right of the debugger window.



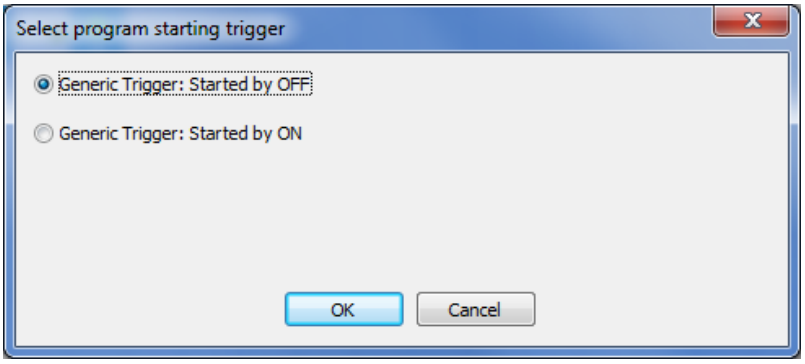
At the bottom of the debugger are seven buttons with somewhat cryptic icons. From left to right they are:

- Start.
  This start the programming running from the current element and execution continues until the program completes or it encounters a breakpoint
- Restart
  Select as the current element the Begin-Here so that you can start over
- Step
  Execute the current element and then stop
- Step Over
  Don't execute the current element. Select as the current element the next element
- Step Over Yes
  Don't execute the current element. Select as the current element the first element on the "Yes" path of a decision element
- Step Over No
  Don't execute the current element. Select as the current element the first element on the "No" path of a decision element

- Open Programs
  Open a dialog showing all the programs in your design. You can select one of those programs and it is opened in its own window. You might want to do this to view the program or to set breakpoints in it.

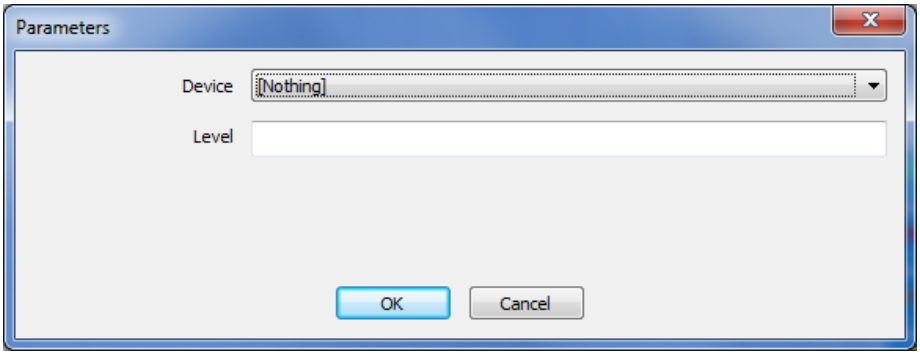
2: Choosing a starting trigger and parameter values

When starting a program running from its Begin-Here element, if the program has triggers or it has parameters, the debugger prompts you to select the starting trigger and/or parameter values.



This program has two triggers. This style of trigger selection dialog shows up to 4 triggers. For programs with more than 4 triggers another style of dialog is used with a dropdown for choosing the trigger.

For programs with parameters, the debugger has you select vales for them when starting the program.

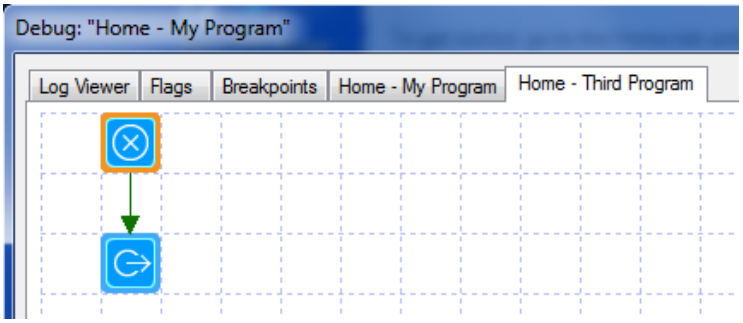


This program has two parameters – the first is an object and the second is a value. Each parameter selection/edit control shows the parameter name to its left. Object parameters have a dropdown showing the objects in your design for you to select from. Value parameters are entered as text and the debugger converts it to a value.

3: Executing elements

Once the program starts you can use the buttons in the debugger control area to control execution. You can use the “step” buttons to move from one element to the next and in that way, see the flow of your program element by element.

The “step over” buttons don’t execute the current element but move the current element to the next element., If it is a decision element you can use the “step over yes” or “step over no” buttons to follow the “yes” or “no” path when you step a decision element.



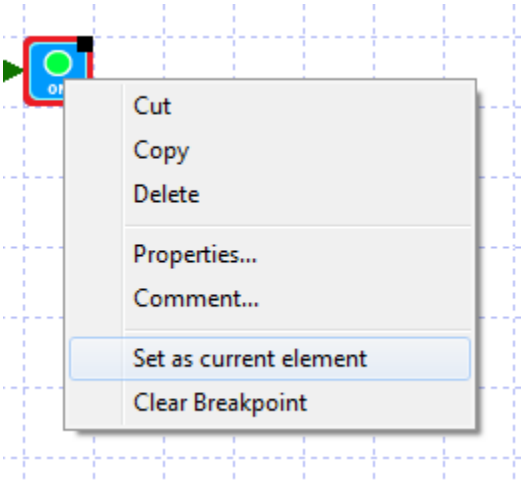
If the program being debugged starts another program using the Start-Program element, and if you “step into” that program or the program stops because of a breakpoint in that program, the debugger shows a tab for that other program in addition to the program that started it. As you can see in the picture to the left, in this new program tab the current element shows with an orange border.

If you were to choose the other program tab – labeled “Home – My Program” in this example - you can see that it also has an element with an orange border. That element is the Start-Program element that started the program shown in the tab to its right. This makes it possible to always know what element your program will come back to when the program started by Start-Program completes.

Note: This is the one place where the element with the orange border has already been executed. As described above, normally the element with the orange border is the element about to be executed. This is done this way so you can quickly find the Start-Program element when moving from program tab to program tab.

Technical Tip: If you are more of a traditional programmer type you can quickly see that the tabs of the programs show the execution stack. The right-most is the program at the top of the stack and the first program from the left end is at the bottom of the stack.

When there is more than one program tab, only the buttons in the debugger control area are active for the right-most program tab since that is where execution is happening.



In addition to use the Start, Restart, and Step buttons which all work with the current element, you can also change the current element.

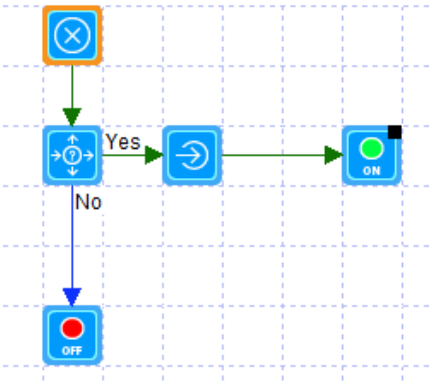
Right-click on an element and select *Set as current element* from the popup menu. This is an easy way to choose the element to next run from or step from.

Choosing the current element and using the Step button is a simple way to execute any element of your choice.

4: Setting breakpoints

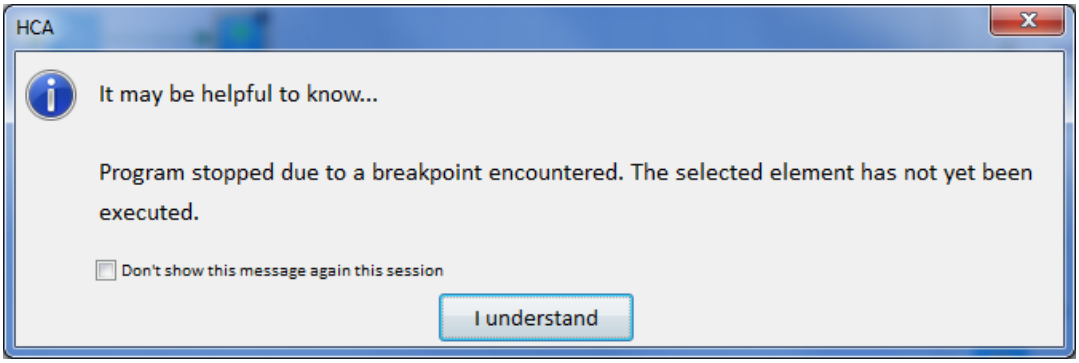
While the "step" buttons are useful you can also mark an element so that before it executes, the debugger stops. These are called "breakpoints". The idea is that you mark certain elements in your program so that when execution gets to them the program stops and the debugger takes over.

To set a breakpoint on an element right click on that element and choose *Set Breakpoint* from the popup menu. If the element already has a breakpoint the menu choice instead shows *Clear Breakpoint*.



When an element has a breakpoint, it shows with a black square in its upper right.

When execution stops at a breakpoint HCA displays a popup:



5: Controlling Breakpoints

The right-most button in the debugger control area opens a selector for all programs in your design. This can be a handy tool when setting breakpoints. Suppose that your program contains Start-Program elements and you want to set breakpoints in the programs that will be started when those elements execute. Use this button to open the list of programs, find the program and open it, and then set a breakpoint in it.

Also in the debugger is a tab that shows all the breakpoints you have set and if they are enabled or not. When created, a breakpoint is initially enabled but can also be disabled. A breakpoint that is disabled remains set as part of the element state but when that element is executed the debugger doesn't stop.

Why would you want to disable a breakpoint? There may be situations where you want to run the program to completion and not stop but not lose the breakpoints you have already set in case the program isn't working as you expect it should. If you delete the breakpoints you may find you need to go back and set them all again.

Note: A breakpoint is part of the element state so it is stored with the element in the HCA file. In this way, you can set breakpoints and they persist between invocations of HCA if you have saved the file.

6: Slow and Fast elements

The debugger divides all elements into two classes: Fast execution and slow execution. Fast elements are one that typically take only a second or two for HCA to execute. Elements like ON, Off, Exit, etc. are all fast. Other elements like the Delay element or elements that request device status can take a longer time to execute. When the debugger encounters an element that is slow, a button appears in the debugger control area.

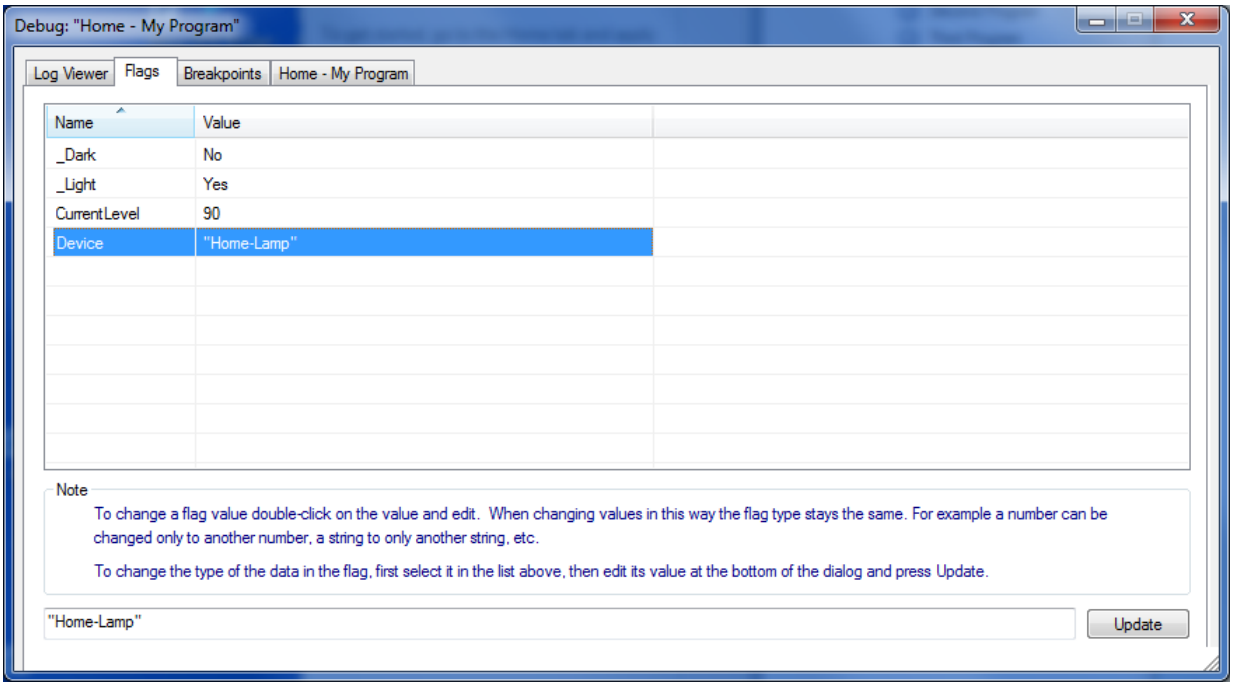
If you press the *stop* button, then execution of the current slow element is aborted and control returns to the debugger. This provides a useful way for a program to have, for example, a delay element set for one hour and not to have to wait for the full hour.

7: Viewing the Log

The first non-program tab in the debugger shows the log. In this log viewer you can choose which of the three logs you want to view, and create, modify, or use log filters. The log viewer in the debugger is provided as a convenience. Alternatively, you could switch back to the main HCA window and use the log viewer there.

8: Viewing and Modifying flag values

The second non-program tab in the debugger shows all the flags ("variables") in your design and their current values. From this tab you can also modify their values if needed. As it says in the tab notes, you can change a flag value by editing the value "in-place" or by using the "Update" button at the bottom.

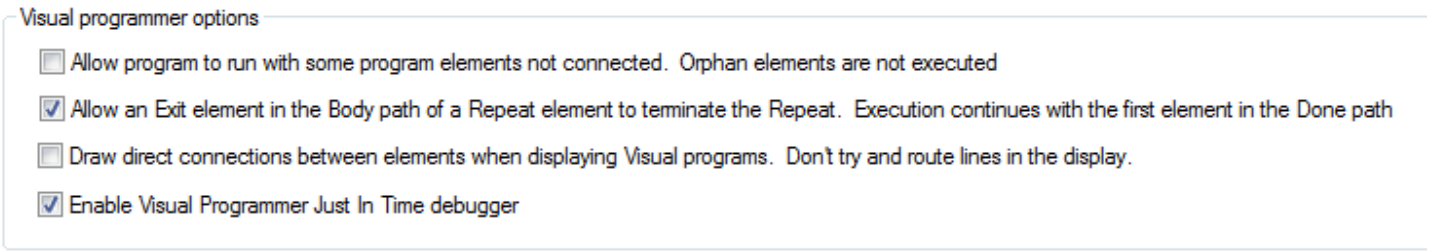


9: Just in time debugging - JIT

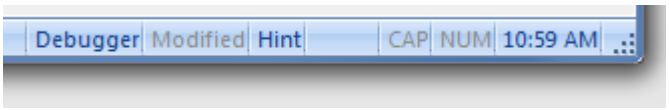
You can use the debugger in two different ways: As described above you can start a program in the debugger, choose it's starting trigger and parameter values, and then step though its execution.

HCA also has what is called "just in time debugging". The idea is that you set breakpoints in programs and then any time HCA during execution of that program encounters an element in a program with a breakpoint, the debugger starts and you can the debug the program from that point on.

This feature must be enabled in HCA Options on the Visual Programmer tab.

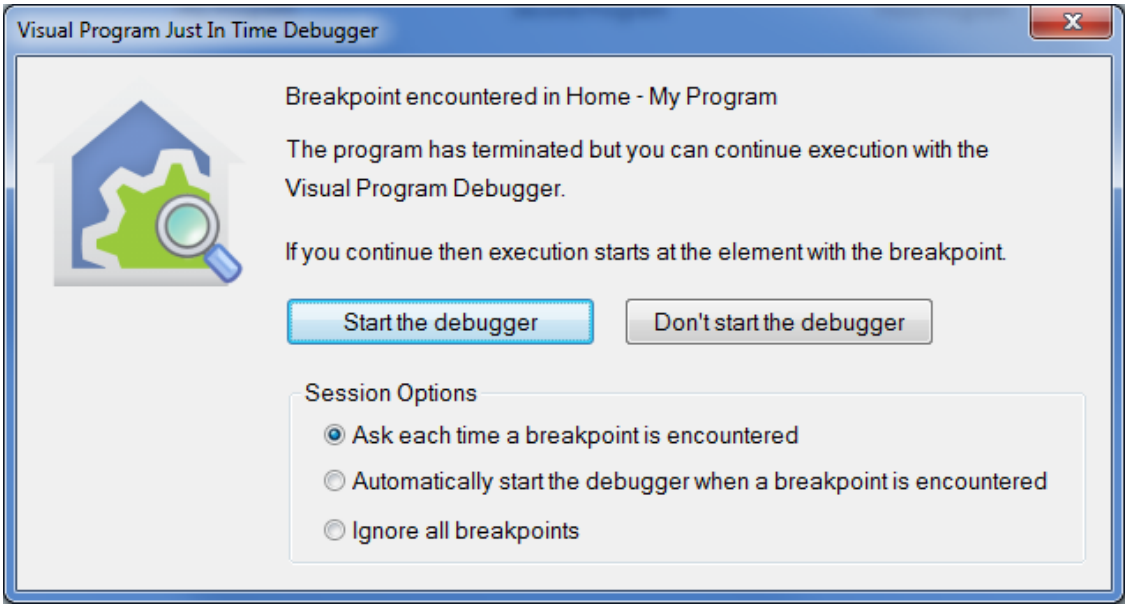


Once JIT is enabled, as a reminder in the main HCA Window status bar the "Debugger" indicator is turned "on".



Once JIT is enabled then any time a program is opened in the Visual Programmer the element popup menu contains the *Set* or *Clear* breakpoint choices. Using these menu choices you can set breakpoints on any element, close the Visual Programmer with OK, and then wait until the program is started by the normal program triggering mechanism. When breakpoints are encountered, the debugger takes over.

If JIT is enabled and an element with a breakpoint is encountered, this popup appears:



This popup tells you the name of the program with the breakpoint and lets you choose to start the debugger or not. It also lets you set options for the next time a breakpoint is encountered. That selection persists until you restart HCA.

If you are currently debugging a program and another program with a breakpoint is encountered, that breakpoint is ignored. Only one program can be in the debugger at a time. If an element has a breakpoint and JIT isn't enabled, then the breakpoint is ignored.

10: Making program changes while debugging

While debugging you can make changes to the program to correct problems. There are some limitations:

- You can only make changes to the program at the "stop of the stack".
- You can't change the Begin-Here element to modify program parameters
- The changes must be saved if you want to continue execution in the debugger. The debugger prompts for this when you begin execution with the *Start* or *Step* buttons.

Tip: It really isn't a good idea to make major changes while in the debugger. If you find you need to make major changes t the program it is a good idea to leave the debugger and then use the normal HCA facilities to edit the program.

##end##