# AC 2009-2159: INTEGRATING ROBOT SIMULATION AND OFF-LINE PROGRAMMING INTO AN INDUSTRIAL ROBOTICS COURSE

**Kevin Devine, Illinois State University**
Dr. Kevin L. Devine is an Assistant Professor in the Department of Technology at Illinois State University. He currently teaches courses in robotics, machining and CNC programming, and solid modeling. Email: kldevin@ilstu.edu.

# Integrating Robot Off-Line Programming and Simulation Into an Industrial Robotics Course

## Background

The importance of robotics in the manufacturing workplace is a given in many industries today. Because of the increased pressures brought on by fierce global competition, it is likely that companies will seek to increase the use of robotics in the foreseeable future[1, 2]. While the automotive industry was the first major industrial base for robotics, new application growth areas of note include such industries as aerospace[3], machining[3, 4, 5], and medical[2]. To support the increased use of robots in a greater variety of workplaces, industry will need experts in the application of robotics[7,8].

Of increasing importance to the users of industrial robots is the use of off-line programming and simulation tools. Systems integrators and end-users of robot systems are finding that the current generation of off-line programming software has a rich set of programming tools that offer great time and cost savings[6]. Today's off-line robot programming and simulation tools offer many advantages to engineering technology programs, making it possible to augment limited hands-on instruction with almost unlimited virtual-robot instruction[7]. It is the author's opinion that modern off-line programming and simulation programs provide many opportunities to improve classroom efficiency and student learning. This paper will (1) briefly describe on-line and off-line robot programming methods; (2) describe how off-line programming and simulation software was successfully integrated into an existing hands-on robotics course at Illinois State University; and (3) discuss the benefits of using these software tools in an educational setting.

## On-Line Robot Programming

Most industrial robots are equipped with a hand-held teach pendant that is used to manually interact with the robot. Traditionally, the teach pendant has been used to create robot programs using the on-line teach method which requires the programmer to manually jog the robot to physical locations in the work cell. These locations are then recorded and text-based program instructions are created to command the robot to move to the recorded locations. Various non-motion instructions are then added to do things such as control program flow (i.e. if-then instructions), increment counters and variables, and to work with inputs and outputs (I/O). Once written, the program is debugged using the teach pendant and physical robot. The on-line programming method has long been the primary means of robot programming. Figure 1 shows a robot and teach pendant.

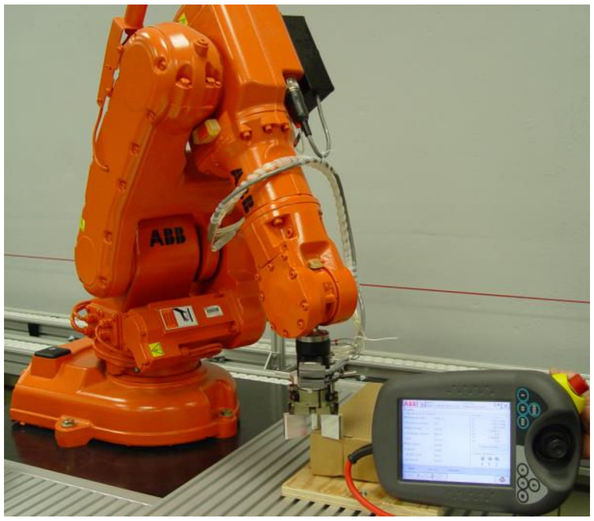


Figure 1. Industrial Robot with Teach Pendant

While still commonly used in industry, on-line programming has many shortcomings. The most obvious drawback is that while the programmer is developing and debugging a program the robot cannot be used for production. On-line programming can also expose the programmer to physical dangers such as hazardous fumes from painting and welding operations and pinch points often found in automated work cells. Also, costly problems with work cell layout, end of arm tooling (EOT) and work holding devices are often not discovered until after they are physically built and the programmer attempts to teach a program[6].

The exclusive use of teach pendants and on-line programming has many drawbacks in instructional settings as well. In many, but not all cases[8], there is no practical way to project the teach pendant screens and menus, presenting logistics problems during instruction. Further, due to cost constraints, most engineering technology programs have very few industrial robots available for instruction, effectively creating an instructional bottleneck when lab activities rely heavily on the use teach pendants. Further complicating matters is the fact that the all too scarce hands-on time using real robots is often used inefficiently because students are not well practiced in the robot procedures they have previously used.

From a class management perspective, relying exclusively on physical robots for instruction makes it difficult to assign meaningful homework and hold students accountable for their learning. Students assigned a programming project, for example, can spend a large amount of class time teaching points and performing simple program debugging tasks such as syntax checking. This makes it difficult to spend time working on more complex issues such as systems integration. Finally, because most teach pendants are small and operated by a single user, it is difficult for students to stay actively engaged when working in groups[8].

## Off-line Robot Programming

Driven by the needs of industrial robot users, robot manufacturers and software vendors have developed a new generation of offline robot programming and simulation tools. Unlike CAM packages that have been used to program and simulate CNC machine tools for decades, off-line programming and simulation systems for industrial robots were not widely used until recent years[6].

Most modern off-line programming and simulation software allows users to import user-created CAD geometry and vendor-supplied robotic components to create a virtual robotic workcell. A virtual robot can be jogged in the virtual workcell to conduct reach studies to verify that the physical robot will be capable of reaching the required locations using the EOT and work holding fixtures as designed. Problems found with the workcell layout can be resolved before the cell is physically built. Some programs allow users to layout an entire plant floor or assembly line while others focus exclusively on developing individual workcells.

Once the workcell layout and EOT designs have been verified, robot programs are developed using a variety of programming tools. Individual robot targets and complete robot paths can easily be created from the imported CAD geometry. A variety of software tools are provided to allow the programmer to make adjustments to robot targets and paths and insert non-motion instructions into the program.

Most software applications provide simulation tools that allow the programmer to simulate robot programs in the virtual workcell. Collision detection, I/O simulators, and cycle time monitors are standard features of many simulation programs. Finished programs are deployed by downloading a text-based program to the physical robot. Downloaded programs are then calibrated from virtual to physical space using user-defined coordinate systems. Because of minor differences between the as-designed virtual workcell and the as-built physical workcell, minor program touch-ups are not uncommon. Once the program is running as desired in the physical world, it may be uploaded back into the off-line programming software, and the virtual workcell can then be modified to match the physical workcell. By modifying the virtual workcell to match the physical workcell, future programs will require less tweaking when downloaded.

There are many compelling reasons why industry is rapidly adopting off-line programming and simulation software. Chief among the benefits is the increased productivity that results from keeping the physical robot in production while programming is underway. Furthermore, problems with inefficient workell layouts and EOT designs can be identified and resolved much earlier and without physical build-up. Programmer safety and job satisfaction is increased because the programmer is no longer required to spend many hours working in the sometimes inhospitable physical work space. Travel expenses can also be greatly reduced because robot programming activities are no longer tied to the physical location of the robot. One systems integrator recently described a project where the robot programs were created in Colorado then emailed to Brazil where they were successfully deployed by a technician after only minor program modifications. The systems integrator estimated it would have taken several months on-site to develop the programs using traditional on-line programming methods.

## Implementing off-line programming in an academic setting

This section describes how off-line programming and simulation software was successfully integrated into an existing hands-on robotics course at Illinois State University. Because the author teaches using ABB robots, this section includes descriptions of functions found in RobotStudio[9], the off-line robot programming and simulation software developed by ABB. Off-line programming and simulation software from other sources will likely have functions similar to those found in RobotStudio.

The initial approach taken was to add new units of study to the end of an existing robotics course. The thought was that it would be helpful for students to have experience programming and operating physical robots before attempting to learn off-line programming. This would allow the students to recognize the application of RobotStudio functions in the real world, making the software easier to learn. Unfortunately this first attempt yielded less than satisfactory results. The most obvious problem with this plan dealt with time constraints and competing class priorities. Because the instruction on off-line programming came at the end of the semester, many students were still struggling to complete other assigned hands-on projects in the class. Furthermore, workloads from final projects assigned in other classes also prevented students from spending quality time working on the RobotStudio lab activities that may have been perceived as being "tacked on" the end of the class. Some students also commented that it was difficult for them to complete the assignments in a timely manner because RobotStudio was only available on-campus. For a variety of reasons, it was evident that students did not spend enough time to work with RobotStudio, and student learning in the area of off-line programming suffered. It was obvious that a new strategy was needed to effectively teach off-line programming in the course.

The next time the course was offered, a revised strategy for teaching off-line programming and simulation was implemented. The revisions revolved around improving student access to RobotStudio, developing more structured RobotStudio learning activities and tutorials, and teaching off-line programming principles in parallel with hands-on robotics instruction.

The first day of the second semester, students were given a copy of RobotStudio to install on their home computer and a RobotStudio lab assignment that was due at the beginning of the next class period. The first lab assignment was very structured, directing students to explore various RobotStudio screens and find definitions and descriptions using the online help files. By requiring students to load and run RobotStudio the first day of class, software installation problems were addressed in a timely manner and the message was sent that RobotStudio was going to be an integral part of the class.

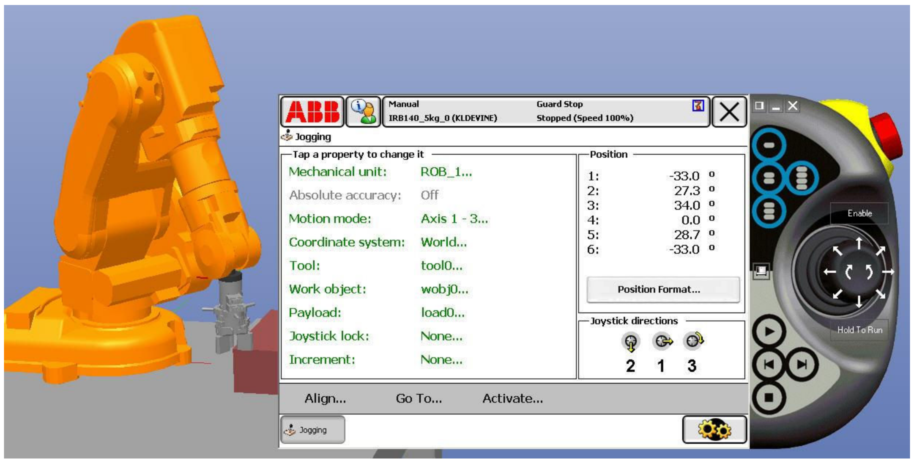
Throughout the course RobotStudio was often used to introduce robot principles before hands-on instruction, rather than after students had worked with physical robots as was done in the previous semester. During most class periods students were given a RobotStudio lab assignment that was to be completed before coming to class the next period. In order to reserve in-class time for hands-on instruction, the RobotStudio labs were designed to be completed at home with minimal in-class time. Rather than giving extensive software demonstrations in-class, brief (2-5 minute) video tutorials were created to accompany each lab. Because each RobotStudio lab required students to submit answers to questions, students generally completed the labs before coming to class. The written work and in-class summary discussions allowed the author to monitor student progress and hold students accountable for keeping up with their RobotStudio work.

In their first RobotStudio lab, students were introduced to the concepts of robot axes and jogging modes by jogging virtual robots at home. Specific procedures and questions accompanying the RobotStudio lab were designed to help students focus on the important concepts for that lab. The premise was that prior student experience with virtual robots would allow the instructor to spend less time introducing concepts in-class, and more time reinforcing and applying previously introduced concepts. As suggested in the literature, deliberate attempts were made to help students make connections between related concepts in the virtual and physical robot environments[10].

As the course progressed, the emphasis shifted away from basic programming and operating principles towards more challenging topics such as program logic/flow and interfacing with peripheral devices. During the later stages of the course, students used RobotStudio to develop their own virtual robot workcell within which they created and debugged robot programs. Ultimately the robot programs were deployed on physical robots using the same procedures used in industry.

## Benefits of using off-line programming and simulation tools in an academic setting

Because RobotStudio uses a virtual teach pendant that is a very accurate replica of the real teach pendant, students were able to interact with virtual robots at home using the same menus, screens and procedures used on a real ABB robot. Figure 2 shows a screen captured image of the virtual teach pendant and virtual robot. The author found the virtual teach pendant very helpful for instruction in a number of ways. First, the virtual teach pendant can easily be projected for all students to see, making in-class demonstrations much more effective. The virtual teach pendant also made it possible for the instructor to develop customized handout materials with screen captured images that exactly match what is seen on the real teach pendant. Furthermore, the virtual teach pendant allowed the instructor to create virtual robot lab assignments that allowed students to program and operate virtual robots outside of class. The virtual labs were designed to provide opportunities for students to practice procedures used in class as well as introduce new procedures and concepts before coming to class. The homework exercises were often designed in such a way as to allow students to learn though experimentation and discovery, which is sometimes problematic for safety reasons when working with real robots. Experience has shown that using the virtual teach pendant outside of class enables students to work more efficiently when working on the real robots.


Manual
IRB140_5kg_0 (KLDEVINE)
Guard Stop
Stopped (Speed 100%)

Jogging

Tap a property to change it

| | |
|---|---|
| Mechanical unit: | ROB_1... |
| Absolute accuracy: | Off |
| Motion mode: | Axis 1 - 3... |
| Coordinate system: | World... |
| Tool: | tool0... |
| Work object: | wobj0... |
| Payload: | load0... |
| Joystick lock: | None... |
| Increment: | None... |

Position

| | |
|---|---|
| 1: | -33.0 ° |
| 2: | 27.3 ° |
| 3: | 34.0 ° |
| 4: | 0.0 ° |
| 5: | 28.7 ° |
| 6: | -33.0 ° |

Figure 2. A virtual teach pendant and robot.

In addition to the virtual teach pendant, RobotStudio has a rich set of graphical programming and simulation tools which have proven to be effective teaching tools. For example, new robot programmers must become familiar with many robot motion control parameters, some of which are difficult to visualize. Program simulations can help students see abstract concepts with clarity. For example, Figures 3a and 3b are a screen images from RobotStudio that illustrate how program motion parameters can affect the path of the robot. Students are able to rotate the virtual robot and watch the virtual robot from several vantage points while the simulation is playing. In this example, the student intended to have the robot weld around the top perimeter of the block as indicated by the red path with straight lines. The black, curved path is the actual path taken by the robot when the program was simulated. Figure 3a illustrates an incorrect robot path. Through experimentation, the student was able to modify a variety of motion parameters to improve the welding process, as shown in figure 3b, and gain a better understanding of the related motion parameters.

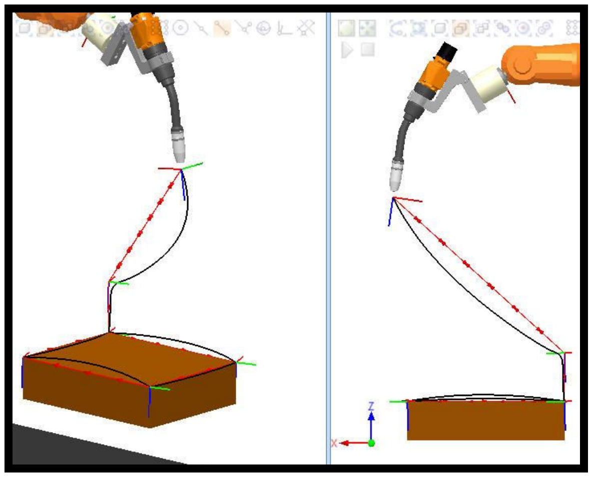


Figure 3a. Incorrect motion parameters

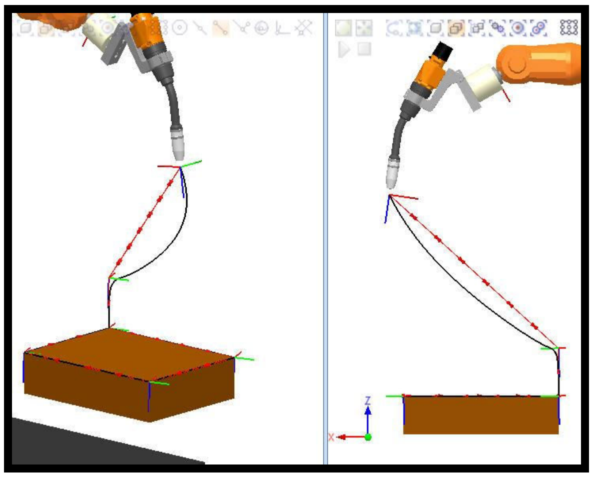


Figure 3b. Correct motion parameters

In another example RobotStudio was used to reinforce their understanding of coordinate systems and help students better understand the procedures used to create them on robots. Figure 4a is an image from RobotStudio illustrating the location of a coordinate system that was created by selecting three points along the edge of a workpiece. RobotStudio was then used to illustrate the coordinate system that would be created by selecting the same three points in a different order (Figure 4b). From this simple exercise, students can clearly see that the order in which the three points are selected affects the orientation of the resulting coordinate system. To help students see the implications of incorrect coordinate system orientation in robot applications, a welding torch was positioned at four robot targets using both the correct and incorrect coordinate systems. As Figures 5a and5b clearly show, the results of an incorrect coordinate system on a physical robot can be quite dramatic.

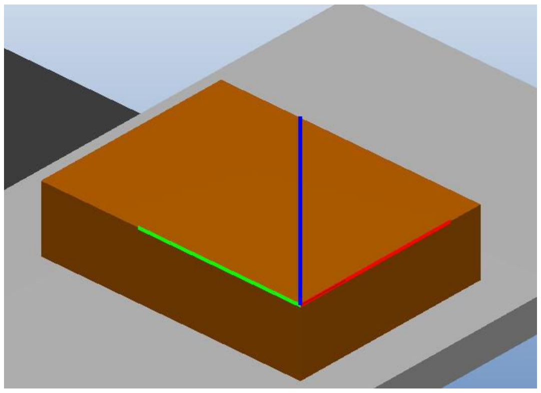


Figure 4a. Correct coordinate system

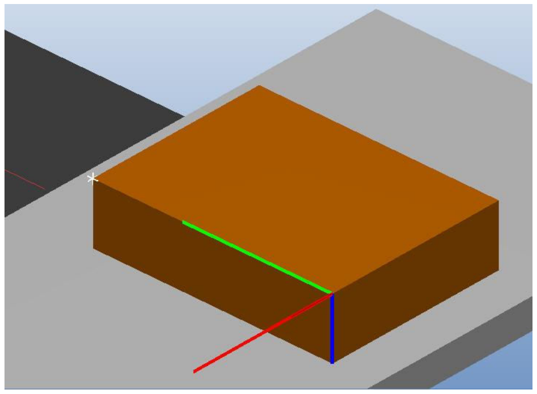


Figure 4b. Incorrect coordinate system

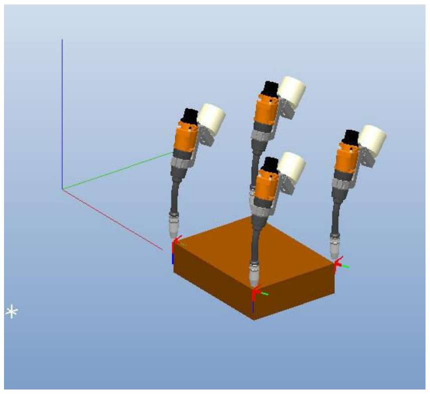


Figure 5a. Tools positioned using correct coordinate system

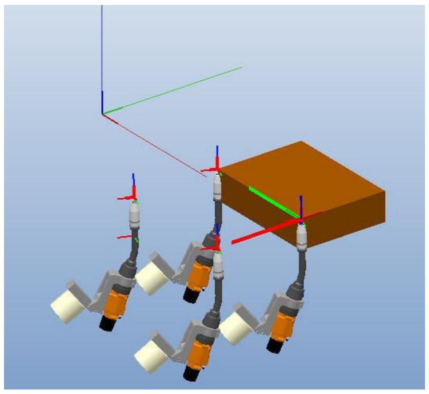
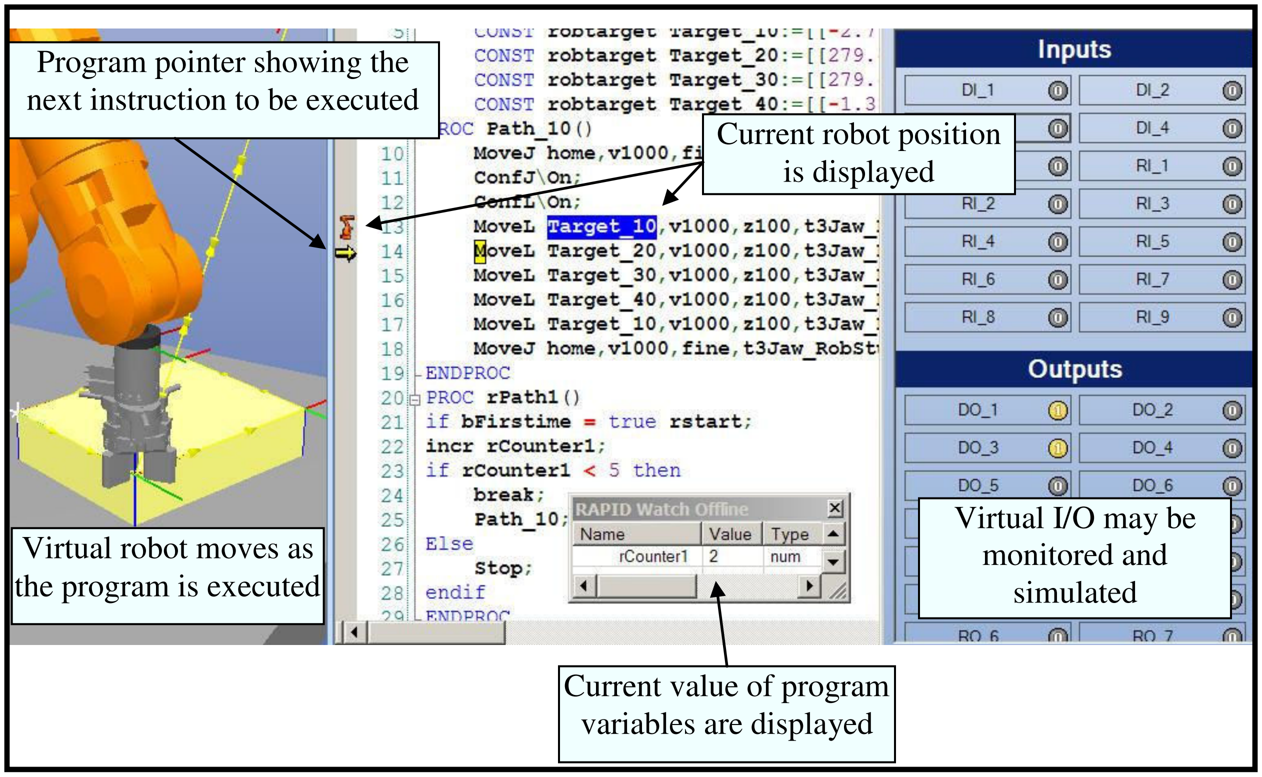


Figure 5b. Tools positioned using incorrect coordinate system

In one final example, RobotStudio proved to be a very effective tool to augment instruction in the areas of program logic/flow. Because most students in the robot course have no prior text-based programming experience, many students struggle to understand program logic. Powerful and essential programming tools such as variables and counters, while-do loops, and if-then statements are challenging for some to learn. The debugging tools built in to RobotStudio proved to be very effective at making these abstract concepts visible. Of great assistance were: the ability to run programs one line at a time, the visual display of the program pointer and current robot location within the robot program code, the variable watch window that shows the current value of program variables, and the I/O simulator that shows the current status of virtual digital outputs and allows users to change the value of virtual digital inputs. Figure 6 illustrates these RobotStudio debugging tools. While at first glance the illustration may look somewhat cluttered, the combination of these tools, along with the real-time movement of the virtual robot, made RobotStudio very helpful during instruction.



```
CONST robtarget Target_10:=[[-2.7
CONST robtarget Target_20:=[[279.
CONST robtarget Target_30:=[[279.
CONST robtarget Target_40:=[[-1.3
ROC Path_10()
    MoveJ home,v1000,fi
    ConfJ\On;
    ConfL\On;
    MoveL Target_10,v1000,z100,t3Jaw_
    MoveL Target_20,v1000,z100,t3Jaw_
    MoveL Target_30,v1000,z100,t3Jaw_
    MoveL Target_40,v1000,z100,t3Jaw_
    MoveL Target_10,v1000,z100,t3Jaw_
    MoveJ home,v1000,fine,t3Jaw_RobSt
ENDPROC
PROC rPath1()
if bFirstime = true rstart;
incr rCounter1;
if rCounter1 < 5 then
    break;
    Path_10;
Else
    Stop;
endif
ENDPROC
```

RAPID Watch Offline

| Name | Value | Type |
|---|---|---|
| rCounter1 | 2 | num |

Program pointer showing the next instruction to be executed
Current robot position is displayed
Inputs
DI_1
DI_2
DI_4
RI_1
RI_2
RI_3
RI_4
RI_5
RI_6
RI_7
RI_8
RI_9
Outputs
DO_1
DO_2
DO_3
DO_4
DO_5
DO_6
RO_6
RO_7
Virtual robot moves as the program is executed
Virtual I/O may be monitored and simulated
Current value of program variables are displayed

Figure 6 RobotStudio debugging tools.

## Conclusion

Although the first attempt to integrate off-line programming and simulation software into a hands-on robot course yielded less than satisfactory results, the second attempt proved to be very successful. Without question, the revised instructional strategy was more effective than the first when it came to teaching students about off-line programming and simulation techniques. Although this was the primary goal of the revision, it was not the only benefit observed. Throughout the entire course, students demonstrated an improved understanding of robot programming and operating concepts and were able to complete hands-on lab activities more efficiently and with greater success. The questions asked by the students throughout the course, performance on written and performance assessment measures, and the quality and complexity of student projects using both virtual and physical robots, clearly demonstrated that the use of off-line programming and simulation software had a positive impact on student learning.

## Bibliography

1. Schneider, R. (2005). Robotic Automation Can Cut Costs. Manufacturing Engineering. Vol. 135 No. 6.
2. Jones, T. (2006). Trends and Motivations for Robot Purchases. www.robotics.org, posted 11/06/2006.
3. Morey, B. (2007). Robotics Seeks Its Role in Aerospace. Manufacturing Engineering. Vol. 139 No. 4.

4. Nieves, E. (2005). Robots: More Capable, Still Flexible. Manufacturing Engineering. Vol. 134 No. 5.
5. Tolinski, R. (2006). Robots Step Up to Machining. Manufacturing Engineering. Vol. 137 No. 3.
6. Robotic Industries Association. (2008). Reduce Start-up Costs with Off-line Programming, www.robotics.org, posted 10/01/2008.
7. Devine, K., Reifschneider, L. (2009). Agile Robotic Work Cells for Teaching Manufacturing Engineering. Proceedings from the 2009 American Society for Engineering Education Illinois/Indiana Section Conference, Valparaiso, IN.
8. Stienecker, A. 2008. Applied industrial robotics: a paradigm shift. In Proceedings, American Society for Engineering Education Annual Conference. Pittsburgh, PA.
9. www.abb.com/robotics
10. Devine, K. 2006. Improving the knowledge transfer skills of industrial technology students. Journal of Industrial Technology, 22(2), 1-10.