THESIS

RAPID DEVELOPMENT OF ROBOTIC CONCEPT STATIONS IN ABB ROBOTSTUDIO

Submitted by

Joseph R. Kopacz

Department of Mechanical Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2013

Master's Committee:

Advisor: David Alciatore

John Williams
Tony Maciejewski

ABSTRACT

RAPID DEVELOPMENT OF ROBOTIC CONCEPT STATIONS IN ABB ROBOTSTUDIO

The purpose of this thesis was to facilitate the design of various tools for Wolf Robotics to support the rapid development of concept stations. Feasibility of new parts is an integral part of design of robotic welding cells and requires a concept station. Many independent developers use virtual simulation to mitigate the risks involved when working with new parts in robotic systems. These developers desire a quick and accurate program to mock up robotic stations equipped with various welding accessories and robot positioners. The current software used by these developers requires advanced training and a large time investment. For these reasons, a quick and accurate way to create new concept cells was created. This was accomplished through the development of programmed add-ins and smart components in ABB RobotStudio. In addition to the smart component series and the add-in, user and maintenance manuals were also developed to train personnel quickly.

TABLE OF CONTENTS

LIST OF FIGURES

DEFINITION OF TERMS

Robot Station – Robot work area including part positioners, safety equipment, robot positioner, and part staging if applicable.

ABB RobotStudio – Application to create a virtual representation of a robot station; used in simulation and offline programming.

ABB PC SDK – A Personal Computer Software Development Kit which allows operators to manipulate the station and objects inside of RobotStudio from custom applications.

IRC5 – Fifth generation ABB robot controller.

Virtual Controller – Virtual representation of IRC5 controller that controls a real world robot.

World Coordinate System – Base coordinate system of an ABB RobotStudio Station defined at X = 0; Y = 0; Z = 0; Rx = 0; Ry = 0; Rz = 0.

Tool Center Point – Describes the position and orientation of the tool. If no tool is connected, it is defined at the center of the flange on the last joint of the robot.

Mechanism – System of moving parts that can be controlled by a controller both virtually and in the real world.

Smart Component – Component, with the ability to manipulate the station and process variables, added to a RobotStudio station through the use of a library.

Add-in – Component, built into RobotStudio, with the ability to manipulate the station and process variables.

Concept Cell – A robotic station built to handle various parts and various processes.

I – INTRODUCTION

To increase the speed of offline programming, a series of smart components and add-ins was created. Each smart component contained the necessary geometry to create the graphical component. The smart components have the ability to create and modify mechanisms so the graphical components will be visually representative of the function performed by the mechanism. The smart components also have the capability of loading non-essential geometry and attaching it to the proper frame in order to make the station as accurate as possible during reach studies and offline programming. Each smart component was programmed in Microsoft Visual Studio C# and was interfaced with RobotStudio through the visual programming enabled by the smart component creator. Variables passed between the station and the smart component .dll were compiled into the program via an .xml document.

An add-in was created in Microsoft Visual Studio C# to allow a user to quickly load a positioner and the desired riser. The positioner is created in the correct position for calibration of a virtual controller. The Head Stock and Tail Stock are selected based on the desired capacity and riser height; the option for a Head Stock and Tail Stock combination with lift is also available. SkyHooks are selected based on capacity, drop, and throw. The appropriate riser is loaded based on the larger of the drop and throw measurements; this is true only when a SkyHook with lift has not been selected. When a Drop Center is loaded, the user is given the option to select a riser.

The smart component series is made of three parts. The first is the automatic fence builder, which allows a user to select the desired length and height of the fence, and build it in the station. The second smart component is the track builder. This smart component loads the desired length of track, gantry based on selected height, and boom length. The user also has the option to load in accessories and have them automatically placed and attached to the proper position on the gantry. This smart component also compiles the gantry mechanism from the height of the tower selected and the travel track length.

The final smart component is the automatic light curtain component. This component allows a user to create multiple light curtains and automatically build a virtual laser fence that sits between the two light curtains posts.

The Automatic Station Builder component was then constructed to leverage the previously developed smart components. This component is given the weight, length, and diameter of a part and autonomously constructs a robotic welding station. Each station is constructed from the following parts: fencing, light curtain, robot positioner, and part positioners. Each component is properly placed based on clearance standards used at Wolf Robotics.

Each smart component has an individual user manual. The user manual covers the installation and usage of each component. The user manuals were created in an effort to expedite the learning process of the new components, although they are fairly intuitive. Maintenance manuals have also been generated to support updates to the smart components throughout its life cycle. A user manual for the add-in was also created; however, no maintenance manual was generated due to the complexity of building the add-in. While no maintenance manual has been compiled, libraries can be updated to newer cad models as long as no major dimensions are changed.

I.1 – Problem Statement

The goal is to generate a toolkit for station developers to quickly and accurately create new concept cells. This includes the desired welding equipment; accessories attached to the appropriate frames of the robot positioner; safety equipment, including light curtains and fencing with arc flash protection; robot positioners with three axis booms of various heights, boom lengths, and lengths of travel track; and part positioners including SkyHooks, SkyHooks with lift, Head Stock Tail Stock combinations, Head Stock Tail Stock combination with lift, and Drop Centers.

I.2 – Motivation

To stay competitive in the market, companies are looking for an accurate and efficient way to load desired models and build positioner mechanisms. When new parts are encountered, a concept station is needed to determine the feasibility of custom customer parts. Many independent developers use virtual simulation to mitigate the risks involved when working with new parts in robotic systems. These developers desire a quick and accurate program to mock up robotic stations including various welding accessories and robot positioners. Currently positioners are human built from individual computer aided drafting models and are compiled into mechanisms allowing a station to simulate kinematics. Other components are loaded into the station as plain geometry and are then colored to resemble the actual component. Due to the extensive list of available positioner geometries and miscellaneous components, a simple way to load the desired models and build the positioner mechanisms is highly desired.

I.3 – Objectives

1. Create a series of smart components to create and update a concept cell quickly and easily.
2. Create a user friendly add-in to build the desired part positioner mechanism based on type, capacity, riser, drop, and throw—where applicable.
3. Utilize the previously developed smart components to autonomously build a robotic welding station given the part weight, diameter, and length.
4. Build accurate mechanisms that can be interfaced with a real controller. This would permit a direct mapping of process paths from a virtual station to a real station.
5. Generate user manuals to expedite training of personnel.
6. Generate maintenance manuals to extend the life of the tools.
7. Illustrate the add-in and smart component series in a case study.

I.4 – Thesis Outline

The need and desire to speed up the virtual robotic programming of a robotic concept station has been covered. This paper will now delve into the research that has already been completed. Various academic sources have developed methods to increase the productivity of individuals using virtual robotic programming. These papers will be discussed in section II.1. Next, the current capabilities and methods used in virtual concept station creation will be covered. This will include the three main programs used in concept cell generation: Motoman, Fanuc, and ABB—this can be found in section II.2. Finally, the current add-ins and smart components that exist to assist in the creation of robotic concept cells will be covered in section II.2. These components cover a wide range of functions and have various degrees of usefulness, but they have all added to the growing knowledge base.

The smart component and add-in series created for Wolf Robotics LLC will be covered in detail in section III. The desired functionality of each component will be discussed in sections III.1, III.2, and III.3. Not only will this cover the desired functionality of the components, but it will also cover the programming methods employed to generate this functionality.

Section IV will present a case study of the generation of a robotic concept station in ABB's RobotStudio. Section IV.1 will detail the development of a robotic concept station without the use of the add-in and smart components. This will show the difficulties that previously existed in the development of concept stations. The smart component series and add-in will then be utilized to build the same station: found in section IV.2. The final section, Section IV.3, will then detail the use of the Automatic Station Builder. This section will show the development of a station using the Automatic Station Builder Smart Component.

An analysis of the success and failures of the currently implemented smart components and add-in will then be discussed in the conclusion—section V. This section includes the statement of results, problems left unsolved, and future works.

The user manuals for the smart components and the add-in are attached in appendices A. General maintenance for the smart components is also attached in appendix B. This covers how the smart components can be accessed and modified for future upgrades. The source code in its entirety can be found in appendices C and D.

I.5 – Acknowledgements

1. Wolf Robotics, LLC
2. Special thanks to Kevin Murphy for giving me the resources to complete this project.

II – BACKGROUND RESEARCH

II.1 – Literature Review

Currently many producers of smart components are private businesses that do not freely give their smart components and add-ins to individuals outside of the company. Given this fact, they also do not publish their works in journals for academic purposes which has made finding academic research of smart components extremely difficult. In lieu of this, a tangent was taken to see what has previously been developed bringing the technology to the current state of smart components and add-ins.

One of the most similar pieces of literature uncovered comes from Chalmers University of Technology. Carl-Johan Rutgersson performed a continuation of previous work to develop a supervisor which would prevent robots within a cell from colliding (1). The first stage of the project performed a virtual simulation of the original program. This simulation was met with some success, but needed to be tested in a real environment. Rutgersson was able to acquire a test cell and program the IRC5 robot controller to manipulate the robots. His tests were successful; however, the reliability of the program was of notable concern.

Another success in the developmental area of robotic cell construction was in the optimal placement of a robot. A paper written by Xiaolong Feng, approached the optimal placement of a robot with a prescribed task in a workspace (2). This program was developed as an add-in to RobotStudio. By analyzing the paths of the robot and placing the robot in the optimal position, they were able to increase productivity by 37%.

More developments in the area of robot programming can also be found in arm programing. Teaching robots simple task has taken a majority of the time of robotic programmers. Unlike todays virtual simulations, robots were programed using teach pendants with a completely built robot cell. To increase the speed of teaching a robot new task, a frame capable of detecting the movements of the

human arm was developed (3). This project was met with limited success but showed the need for a virtual representation of the robotic cells.

ABB has led the way in robotic programming by creating software packages that empower engineers by simplifying the task of robotic programming (4). The creation of RobotStudio has granted individuals with little to no programming experience the ability to create simulations in minutes. This software has drastically shortened the time necessary to program robotic work stations. This has come to a precipice with the addition of user programmed add-ins and smart components.

II.2 – Current Station Creation Software

While a large community of robotic developers exists, there are three major players: ABB, Fanuc, and Motoman. These three developers not only build the robots and controllers, but have also created offline programming tools. They have each created their own offline programming tool to handle the manufacturing processes they support.

ABB has created RobotStudio, which utilizes a modern graphical user interface and a Personal Computer Software Development Kit (PC SDK) that interfaces directly with RobotStudio (5). Increased flexibility of the RobotStudio PC SDK resulted in the creation of independent smart components and add-ins by external developers (6). Figure 1, shows an example station used by Wolf Robotics to simulate the welding of new customer parts.

Fanuc created RoboGuide and a PC SDK that is capable of interfacing directly with a robot or robot neighborhood (7). Motoman developed MotoVisual Create with component configurable plug-and-play (8). This software is similar to what ABB has created, but is not as extensive.

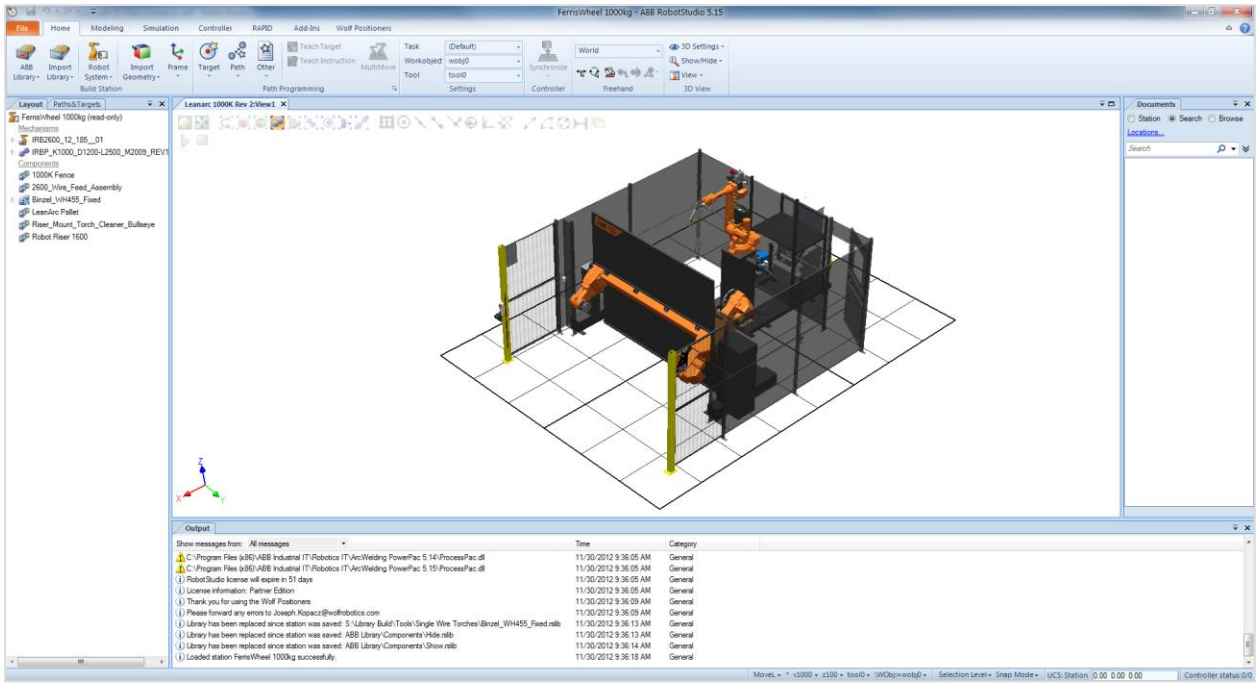


Figure 1: RobotStudio with Example Wolf Lean Arc Station

The software built by these competitors is similar in function but vastly different in execution. First, and arguably the most important, is the cell layout function. It is important to build accurate cell layouts to deliver to a customer for various reasons. In manufacturing, floor space is extremely valuable. It is important to know the exact dimensions of the robotic cell and to ensure appropriate room is inside of the cell for part loading, unloading, and operation. For these reasons, developers of the software have included measurement tools which can take virtual measurements of the cell; this can be valuable in both the virtual layout and during the manufacturing of these cells. It is imperative that the virtual cell matches the actual cell. If a cell is not accurate, virtual paths cannot be taken directly from the virtual cell and compiled into actual robot code for use in a real model.

In the hope of making cell layout faster, the software developers have created standard libraries for use. These include human machine interfaces, part positioners, miscellaneous welding supplies, and safety equipment such as fencing and light curtains. While these libraries are a valuable time saving tool, it is

not always desired to use the supplied components. In this case, it is possible to integrate custom geometry from computer aided drafting software. These files can then be modified in the station using a limited number of geometry modifying tools such as coloring and surface deletion. This function is not only important when laying out the initial cell concept, but also when importing customer geometry—as this geometry is often large and extremely complicated.

To position these libraries inside of a virtual cell, a user can either drag the library to a certain location or set the location via a dialog. When setting the coordinates of a library, it can be set relative to the world coordinate system declared by the station, to a user created frame, or to the base frame of another component. The units of length can be changed between Standard and English, and the units of angles can be set between radians and degrees.

Due to the size and complexity of the customer parts, it is often necessary to place the parts in different orientations for the various manufacturing processes. Different positioners are used to create the desired degrees of freedom needed. Not only do these manufactures provide the robot and controllers, but they also offer standard positioners that can interface directly with the supplied robot controller. This allows for coordinated robotic motion. These positioner libraries come with the offline programming software. If a custom positioner is desired, there are mechanism builders that can be leveraged to create custom positioners. With this functionality, users are able to create custom mechanism built from imported geometry. By selecting the position of the joint and the body it will act on, a prismatic or rotational joint can be created. These positioners can then be saved and used in future stations.

Once a part has been placed into the desired orientation, the next step is to perform a reach study with a robot. Although a part may be inside the reach area of a robot, it is important to make sure the robot tool center point (TCP) can be placed in the correct orientation on a part; for example it can be oriented

perpendicular to a surface for painting. Each company's software has implemented the functionality of reach studies.

The next step of a feasibility study is path planning. This process tells the robot at what angle and offset the robot's TCP needs to be to follow a desired path. This can be accomplished in multiple ways. The first is selecting start points, end points, corners in between, and having the software interpolate a linear path. This can also be performed on circular arc segments by selecting the starting point, ending point, and one other point along the curve; this allows the program to determine the orientation, radius of a circle, and arc length. Many times a simple circle or linear path will not accomplish a specific path. If that is the case edge detection can be used to create the desired path.

The final step in a feasibility study is simulation. In this step, the robot runs off of a virtual controller to ensure all commands can be followed appropriately. For this to succeed, the robot's TCP must always be in a reachable position. The robot also has to be able to complete a move instruction without changing its arm configuration and without passing through singularity. During this process, the station can also perform collision detection, and ensure the robot does not run into the part or collide with any of the components in the station. A simulation can also provide projected process times, and these process times can be used to estimate the amount of time the entire part will take to produce.

II.3 – Current Implementation of Smart Components and Add-ins

With the addition of add-ins and smart components to ABB's RobotStudio many individuals have taken to programming simple smart components. These smart components vary in usefulness, but all add to the knowledge base that is rapidly developing around RobotStudio. The section below covers some of the programs that are pioneering add-ins and smart components.

II.3.A – Parametric Fence

The Parametric Fence smart component, created by Anders Spaak, builds a box fence (9). This smart component allows a user to change the length and direction of any of the fence pieces. Using this smart component also allows the user to change the color of the fencing they have loaded. Figure 2, shows the user interface with the smart component. This figure shows how the user can change the length of any section of fence and set the color. Figure 3 shows the graphical output created in the station by the smart component.

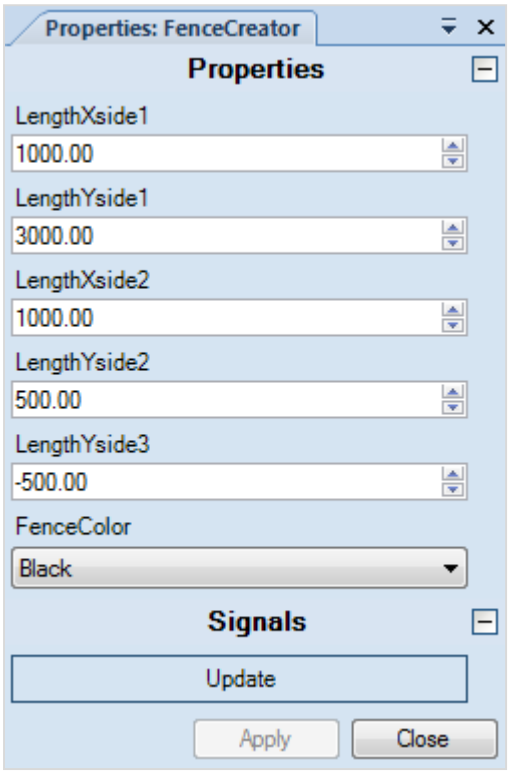


Figure 2: User Interface with Parametric Fence Creator

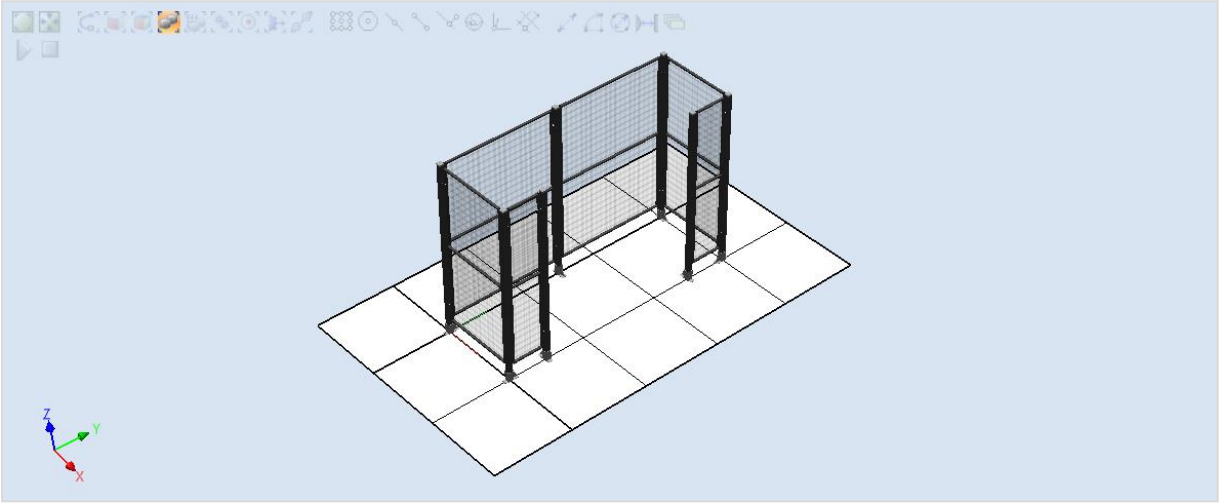


Figure 3: Graphical Output of Parametric Fence Smart Component

II.3.B – Parametric Robot Stand

Parametric Robot Stand, created by Dean Thomas, is a smart component that leverages RobotStudio's ability to create simple geometric bodies and attach them to create a part (10). This smart component allows the user to select the top diameter and height of the stand. As the top diameter is changed, the base and main cylinder size is changed proportional to the top diameter selected. When the height is changed, the main cylinder dimension in the z-direction is changed. This also changes the position of the top plate so it is still sitting on top of the main cylinder. Figure 4 show the user interface generated by the smart component. In this dialog box the user can select the diameter and height and update the robot stand. Figure 5 shows the graphical output created in the station by the Parametric Robot Stand smart component.

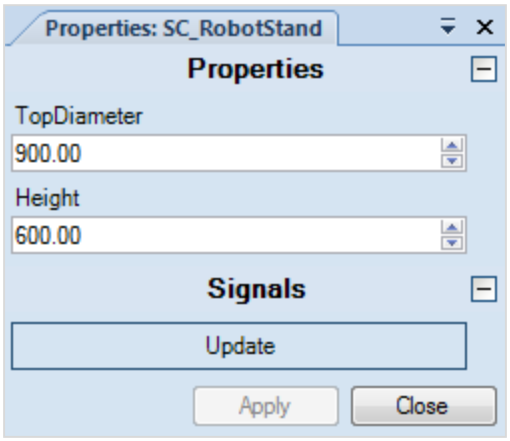


Figure 4: User Interface with Parametric Robot Stand

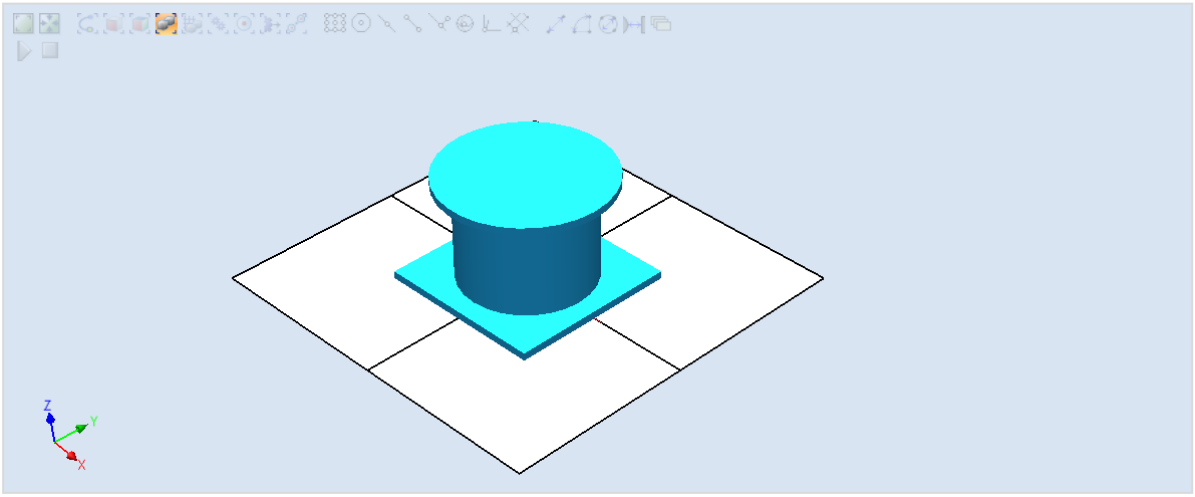


Figure 5: Graphical Output of Parametric Robot Stand Smart Component

II.3.C – Schunk Gripper

Schunk Gripper, by Richard Ramos, is a gripper that will automatically attach and detach a part it senses in its gripper (11). When placing a part in the grippers and selecting the button *doGripper*, the jaws of the gripper will close and the part will move with the gripper. The jaws will open and release the object in its current position once the button *diGripper* is selected, seen in Figure 6. A yellow globe appears next to the current status of the gripper. The output seen in the station is portrayed in Figure 7.

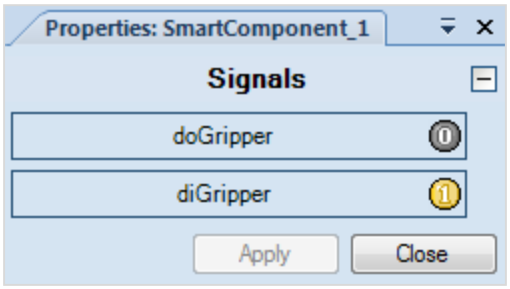


Figure 6: Schunk Gripper Smart Component User Interface

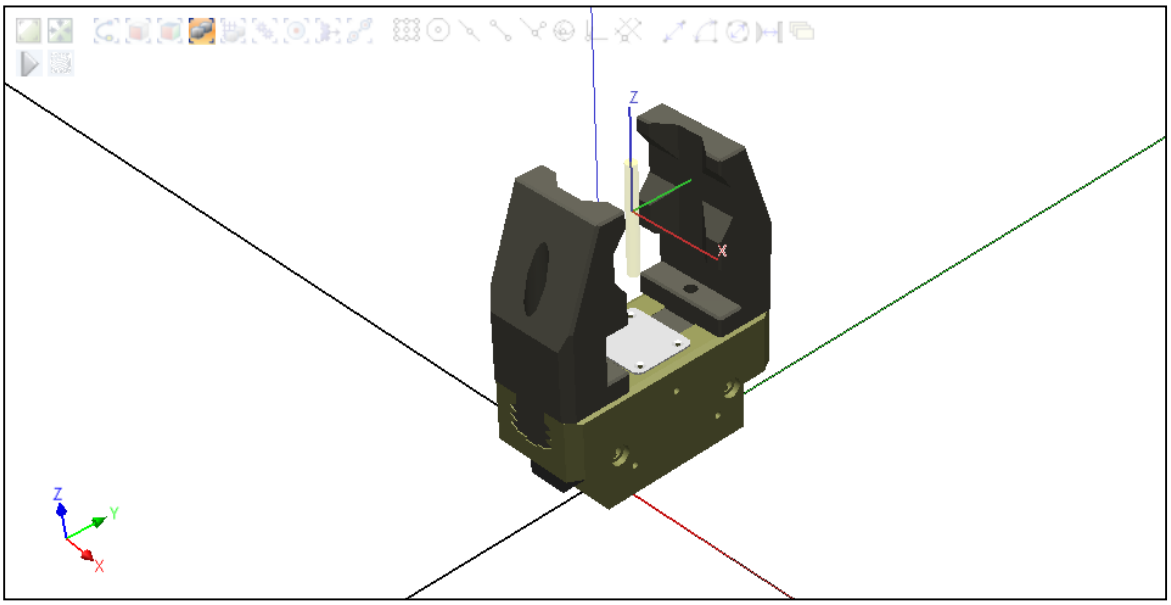


Figure 7: Graphical Representation of Schunk Gripper Smart Component

II.3.D – Coordinate File Import Add-in

The coordinate file import add-in was created by Simon Fogbring (12). This add-in has some bugs in the newest version of RobotStudio (5.14, 5.15). Previously this add-in would allow a user to add targets to a path from a supported file type. This aided in the programming of robotic paths. The supported file formats are nc (Numeric Control), csv (Robcad Export), crd (ABB S3 Coordinate Files), cls, aptsource (Catia Export), and aml (Manz export). This add-in rejects coordinates that are closer than the minimum set distance. It also has the ability to output coordinates when the angle deviated more than the maximum angle; this is relative to the previous coordinate successfully imported. It would automatically create circular instructions if the radius was lower than the Max Radians.

II.3.E – RS User Library Add-in

The RS User Library Add-in was created by ABB to showcase some of the abilities of add-ins and to create an easy-to-access user library (13). This add-in would create a user library button on the home tab. This tab would automatically populate the libraries contained in a certain file structure. The files must be contained in the user projects folder and point to a folder call Libraries. Individual folders under Libraries would create segments in a pop down window. Each segment would include an image of the libraries available under that file structure, seen in Figure 8. This was originally looked at for the Wolf Robotics Libraries; however, due to the recursive geometry and the extent of the Wolf Libraries another route was chosen.

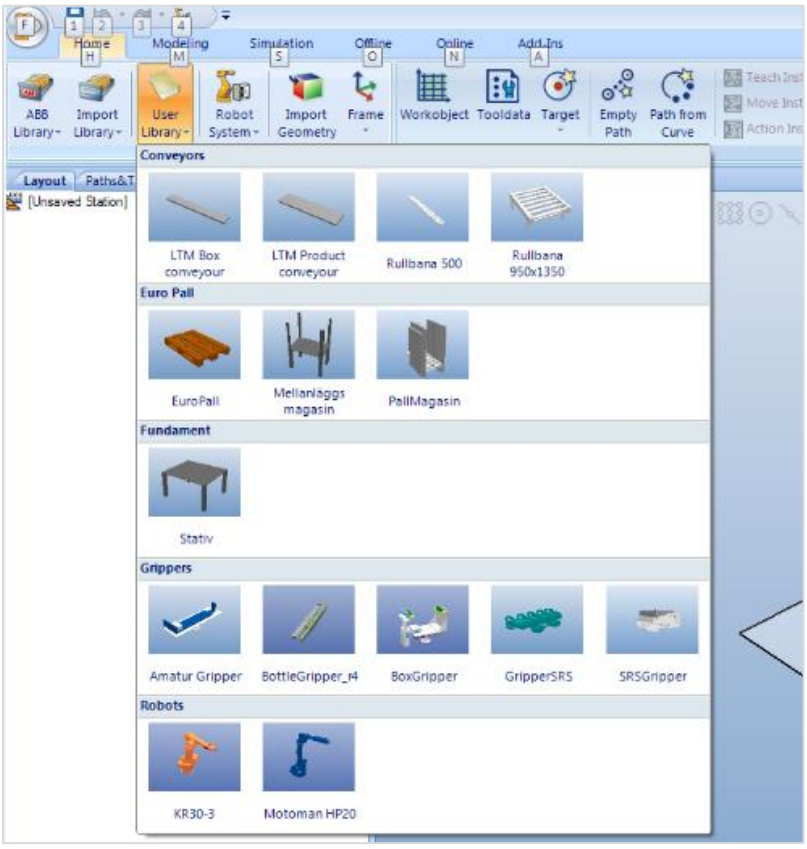


Figure 8: RobotStudio User Library Add-in

III – DISSCUSSION

While the libraries supplied by the developers are useful, custom libraries from each developer are common. This process includes bringing in the desired geometries, coloring the surfaces of the geometry, and building multiple geometries into a mechanism if one is required. ABB's RobotStudio has worked to overcome these short comings by creating a Software Developers Kit. This includes various commands that can manipulate a virtual station given a set of constraints. The SDK natively supports Microsoft's Visual Studio C#; by using the supplied program templates, a debugger can monitor the add-in or smart component while it is running inside of RobotStudio. This allowed for greater ease of error trouble shooting during programming.

Before the smart components and add-in could be created, the Wolf Robotics positioners and parts needed to be converted into RobotStudio libraries. The Wolf Robotics standards were first dissected to determine what major parts made up each component; noting repeating geometries between components. The graphical component libraries were then created from over 500 individual cad files. Each part was carefully positioned in separate libraries and painted the appropriate color. These libraries were then separated into a hierarchical folder structure for the smart components and add-in to load.

III.1 – Smart Components

III.1.A – Automatic Fencing

A component to bring in a selected length of fence was desired so single panels of fence would not have to be individually positioned and oriented. It was also desired that the fence length or height could be easily updated if the station were to change in the future. This component was built to use the RobotStudio component matrix repeater. This would allow for a single object to be repeated linearly along a selected direction thus limiting the number of required attached geometries. An example of this smart component can be found in Figure 9, seen below.

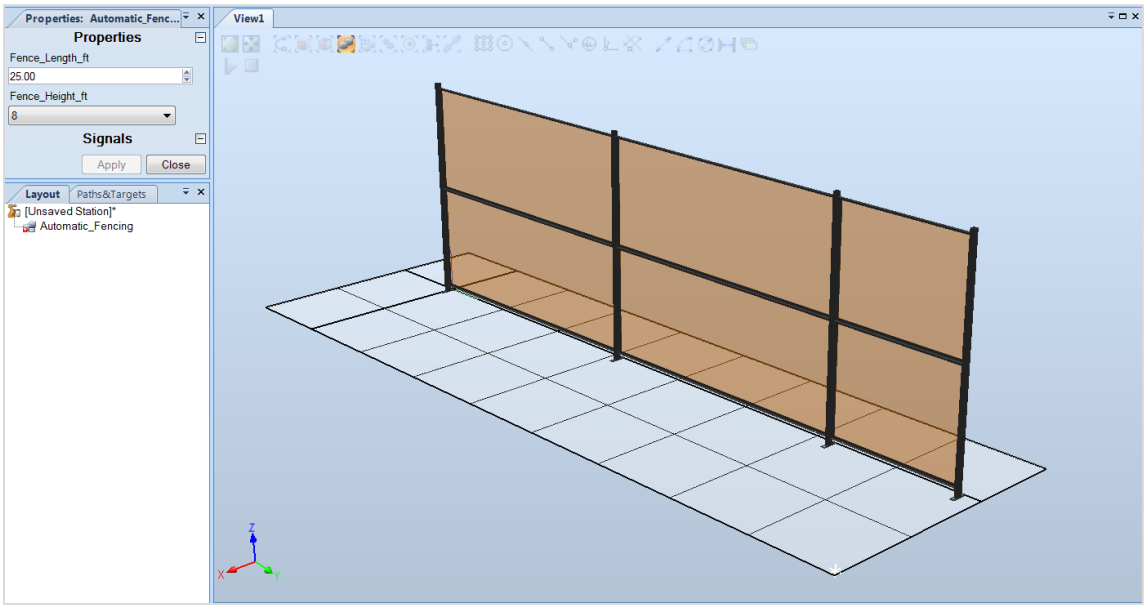


Figure 9: Smart Component Automatic Fencing

The first step was to declare variables in such a way that a user of RobotStudio could set the variables and the station would reflect the change. This was accomplished using the XML file that is created when a RobotStudio add-in is created; this XML file is compiled into the RobotStudio Library (.rslib). A code segment of Fence_Calculator can be seen in Figure 10; this code segment declares one variable that can be read or written to in RobotStudio and three variables that can be read from RobotStudio. The three read-only values are used in the matrix repeater of RobotStudio: count, distance, and offset. *Count* determines the number of 4 foot by 10 foot fence panels necessary, *distance* determines the interval at which these fence segments will be placed, and *offset* determines where the last fence panel should be placed.

```
<DynamicProperty name="Fence_Height_ft" valueType="System.Int32" value="0">
  <Attribute key="Allowed" value="4;8;10;12"/>
</DynamicProperty>

<DynamicProperty name="count_4ft_10ft" valueType="System.Double" value="0" readOnly="true">
  <Attribute key="MinValue" value="0"/>
  <Attribute key="Quantity" value="None"/>
</DynamicProperty>
<DynamicProperty name="dist_4ft_10ft" valueType="System.Double" value="0" readOnly="true">
  <Attribute key="MinValue" value="0"/>
  <Attribute key="Quantity" value="Length"/>
</DynamicProperty>
<DynamicProperty name="offset_4ft" valueType="System.Double" value="0" readOnly="true">
  <Attribute key="MinValue" value="0"/>
  <Attribute key="Quantity" value="Length"/>
</DynamicProperty>
```

Figure 10: Fence Calculator Smart Component XML found in Appendix C.

The next step was to format the variables into a form that could be used in the CodeBehind file for calculations. The CodeBehind file is included in the RobotStudio Library when compiled. Figure 11, is a code segment from Fence_Calculator that reads the value from a RobotStudio station and sets it to an integer that can be used by the SmartComponent.

```
int Orig_Fence_Height = (int)component.Properties["Fence_Height_ft"].Value;
```

Figure 11: Fence Calculator CodeBehind found in Appendix C.

Finally, the number of 10 foot segments is calculated using a simple algorithm. The fence length is divided by each integer from ten to one and the whole number is saved in an array, by truncating the remainder. The above described code can be found in Figure 12.

```
fence[9] = Orig_fencelength / 10;
fence[9] = (int)Math.Floor(fence[9]);
temp = Orig_fencelength % 10;
x--;

while (x >= 0)
    {
        fence[x] = temp / (x + 1);
        fence[x] = (int)Math.Floor(fence[x]);
        temp = temp % (x + 1);
        x--;
    }
```

Figure 12: Fence Calculator CodeBehind found in Appendix C.

The array created by the algorithm is then used to determine the necessary offset and number of segments needed. The distance between fence panels (offset) is known from the length of the fence plus the length of a support pole. Old values are then cleared so that if a panel is updated, incorrect segments will not be created. This was implemented in C#, a segment of code showing this can be found below in Figure 13.

```
        component.Properties["count_4ft_10ft"].Value = (fence[9]);
        component.Properties["dist_4ft_10ft"].Value = (3.0988);
        component.Properties["offset_4ft"].Value = (fence[9] * 3.0988);

        component.Properties["count_8ft_10ft"].Value = (0);
        component.Properties["dist_8ft_10ft"].Value = (0);
        component.Properties["offset_8ft"].Value = (0);
        component.Properties["count_10ft_10ft"].Value = (0);
        component.Properties["dist_10ft_10ft"].Value = (0);
        component.Properties["offset_10ft"].Value = (0);
        component.Properties["count_12ft_10ft"].Value = (0);
        component.Properties["dist_12ft_10ft"].Value = (0);
        component.Properties["offset_12ft"].Value = (0);
```

Figure 13: Fence Calculator CodeBehind found in Appendix C.

Once the CodeBehind has been compiled it can be loaded into RobotStudio and leveraged by the RobotStudio visual programing. Figure 14, shows how the Fence_Calculator code interfaces with the visually programmed smart component. Visual programing uses wires to set variables equal to each other. Here it can be seen that the variables Fence_Length_ft and Fence_Height_ft, defined by the user in the station, flow into the Fence_Calculator smart component; this is what was created above in C#. The output from this smart component then flows to additional smart component called 4ft, 8ft, 10ft, and 12ft.

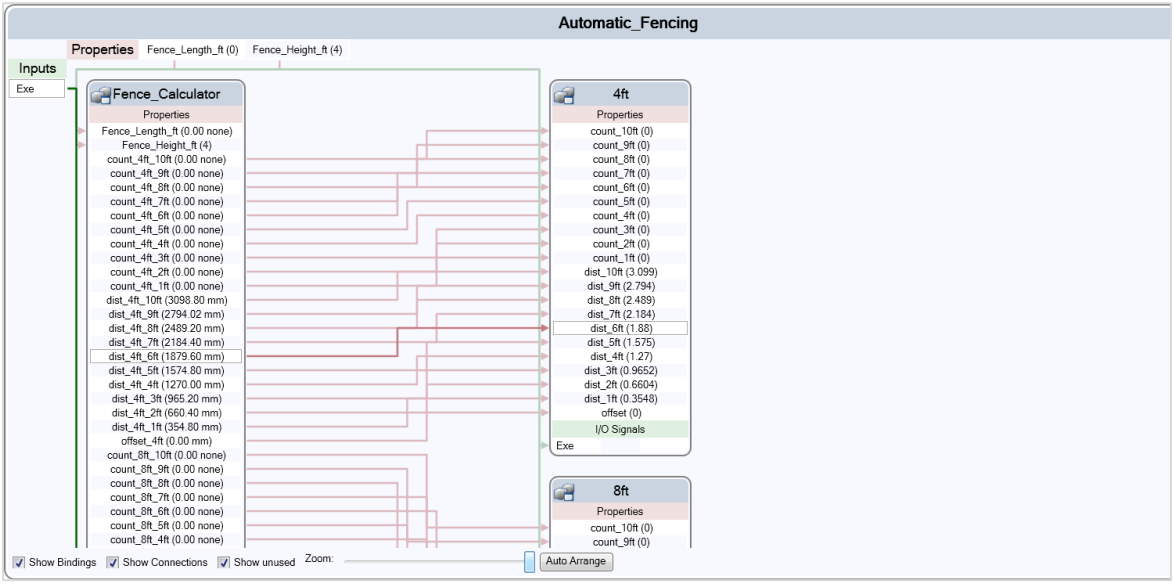


Figure 14: Fence Calculator RobotStudio found in Appendix C.

The smart component for fence heights of four feet can be seen below in Figure 15. This smart component contains the hide component, the matrix repeater, and the transformation expression. The hide component is responsible for hiding the root instance of the graphic component—in this case a single length of four foot fence. There is one hide component for each of the 10 standard lengths used. This is refreshed every time a change is made to the main component; this ensures undesired fence segments are hidden from the users view. The matrix repeater is responsible for creating the appropriate number of fence segments and offsetting them the appropriate distance. This component is also linked to the root graphic component so the correct geometry is duplicated. The final component was created as an expression, this expression translates the matrix repeater; this allows multiple fence lengths to be used at once. Three additional components were also created to control the eight foot fence segments, ten foot fence segments, and twelve foot fence segments; these can be found in Appendix C Fence_Calculator RobotStudio.

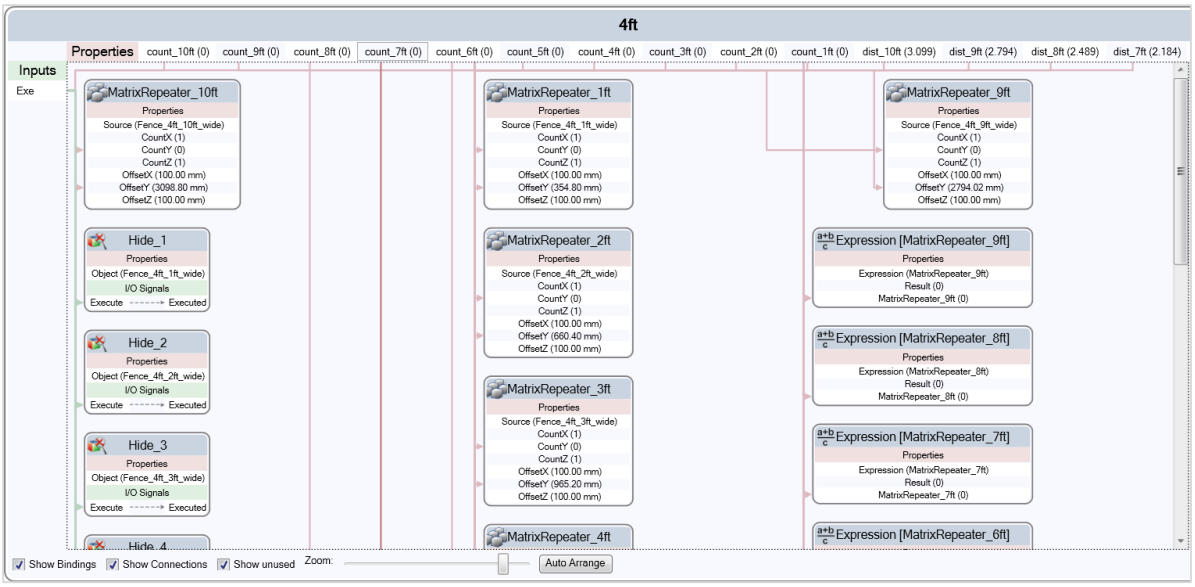


Figure 15: Fence Calculator RobotStudio found in Appendix C.

III.1.B – Automatic Track Builder

To simplify the process of bringing in a robot travel track, boom, and necessary welding equipment, a component was created to load the necessary geometry and update an existing mechanism. This component's joint limits would need to be representative of the length of the track and the height of the tower. This can be found in the figure below, Figure 16. The first step in this process was to look at the standard Wolf Travel Tracks. There are 20 standard sizes that can be used; each contains its own combination of pieces. Appendix C. Track_Builder Segment Order shows the part order that must be used to generate the different size tracks.

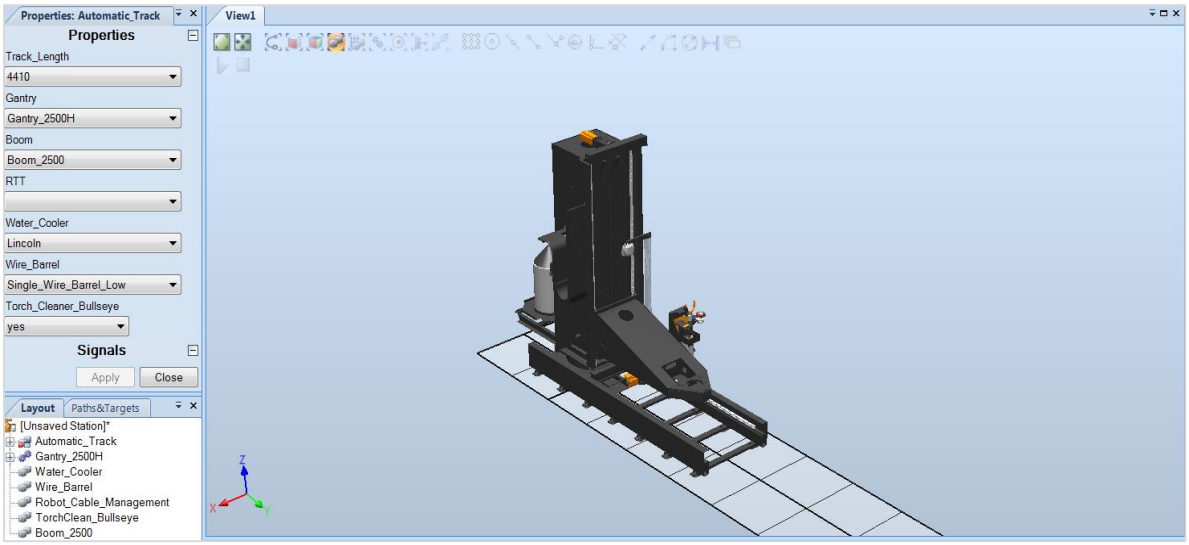


Figure 16: Smart Component Automatic Track Builder

Initially, variables are created and loaded in a manner similar to the method used in Automatic Fencing. A mechanism representative of the standard track lengths used at Wolf Robotics was desired so the track in the virtual station is built to specifications outlined in the tables found in Appendix C. Fence Calculator Segment Order. Large arrays were built to store the number of each segment used in a given length of track. Next, the Track_Builder component looks for previous geometry in the station and deletes it so new components are not created on top of old ones. This method of loading current station components and deleting them is used in many of the smart components and is seen in Figure 17.

```
GraphicComponent temp;
       Station station = Project.ActiveProject as Station;
       station.GraphicComponents.TryGetGraphicComponent("Carriage", out temp);
       if (temp != null)
       {
           station.GraphicComponents.Remove(temp);
       }
```

Figure 17: Automatic Track Builder CodeBehind found in Appendix C.

After the old geometry has been deleted from the station, a new mechanism is then loaded into memory based on the desired tower height. Joint limits are then set to the length of track that has been

built. The new mechanism is then added to the station and disconnected from the library so future changes to the mechanism are not saved in the mechanism library. This process is seen in Figure 18.

```
        //Load mechanism tower
        Station station = Project.ActiveProject as Station;
        GraphicComponentLibrary mylib;
        mylib =
GraphicComponentLibrary.Load("S:\\LibraryBuild\\Custom_Lib\\Gantry\\Gantry_2500H.rslib", true);
        Mechanism Gantry = (Mechanism)mylib.RootComponent.CopyInstance();
        Gantry.Name = "Gantry_2500H";
        Gantry.SetJointLimits(0, 0, ((Orig_Track_Length - 1667.05) / 1000));
        station.GraphicComponents.Add(Gantry);
        Gantry.DisconnectFromLibrary();
```

Figure 18: Automatic Track Builder CodeBehind found in Appendix C.

The selected boom length is then loaded into the station and placed relative to the tower that was added to the station. The Gantry mechanism is then dissected to retrieve the proper mechanism joint. The boom is connected to this link with a matrix describing its orientation relative to the world coordinate system—this can be seen in Figure 19.

```
        mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_1500.rslib",true);
        Part Boom = (Part)mylib.RootComponent.CopyInstance();
        Boom.Name = "Boom_1500";
        station.GraphicComponents.Add(Boom);

        //Place boom in correct position
        Boom.Transform.X = .708271;
        Boom.Transform.Y = .666345;
        Boom.Transform.Z = .946033;

        //Attach boom to mechanism
        GraphicComponent L4;
        Gantry.GetParentLink(3, out L4);
        Part temp = (Part)L4;
        temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));
```

Figure 19: Automatic Track Builder CodeBehind found in Appendix C.

Again the CodeBehind is compiled and loaded into RobotStudio so that the RobotStudio visual programing can leverage the newly created smart component. In this smart component the user defined variables from the station are fed into the Track_Builder smart component via wires. This component then preforms the calculations described above and feeds them into the four parts that make up the track: Rail_Weldment, Rail_Cover, Rack, and Star_Rail—seen in Figure 20.

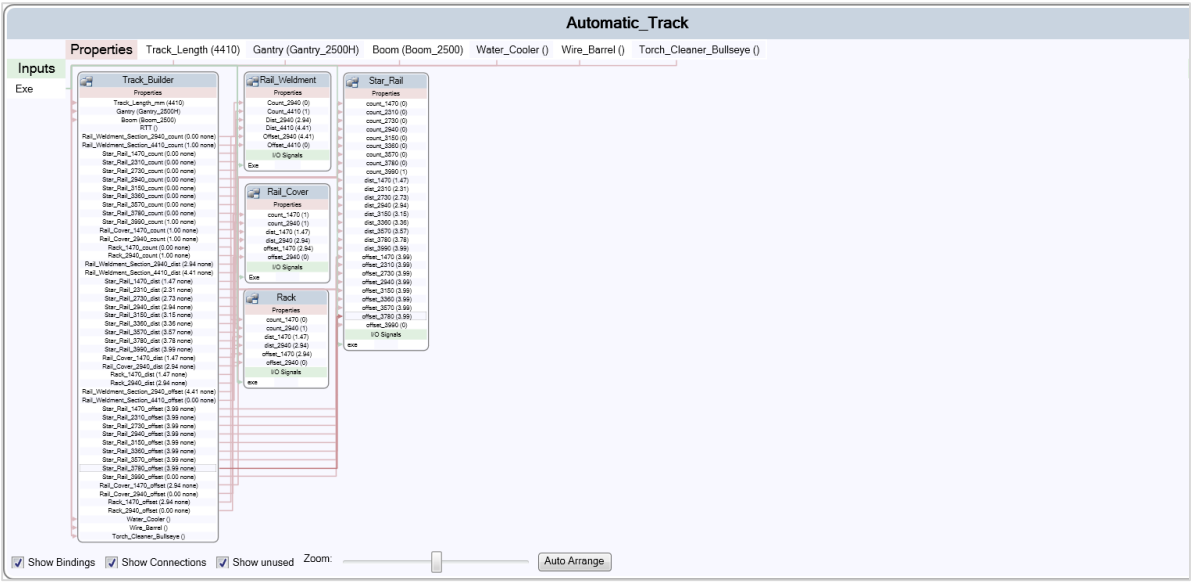


Figure 20: Automatic Track Builder RobotStudio found in Appendix C.

To give a general idea of how each of these smart components function the Rail_Weldment component will be discussed in detail. The visual programing of this smart component can be seen in Figure 21. This shows the three major components of this smart component: the hide component, matrix repeater, and translation expression. Like before the hide component removes the display of the root graphic component seen in the station. The matrix repeater is responsible for creating visible instances of the hidden component when they are appropriate. The matrix repeater is also responsible for the offset of each rail. The translation of the matrix repeater is controlled by an expression. This expression takes the offset given from the Track_Builder component and translates the matrix repeater.

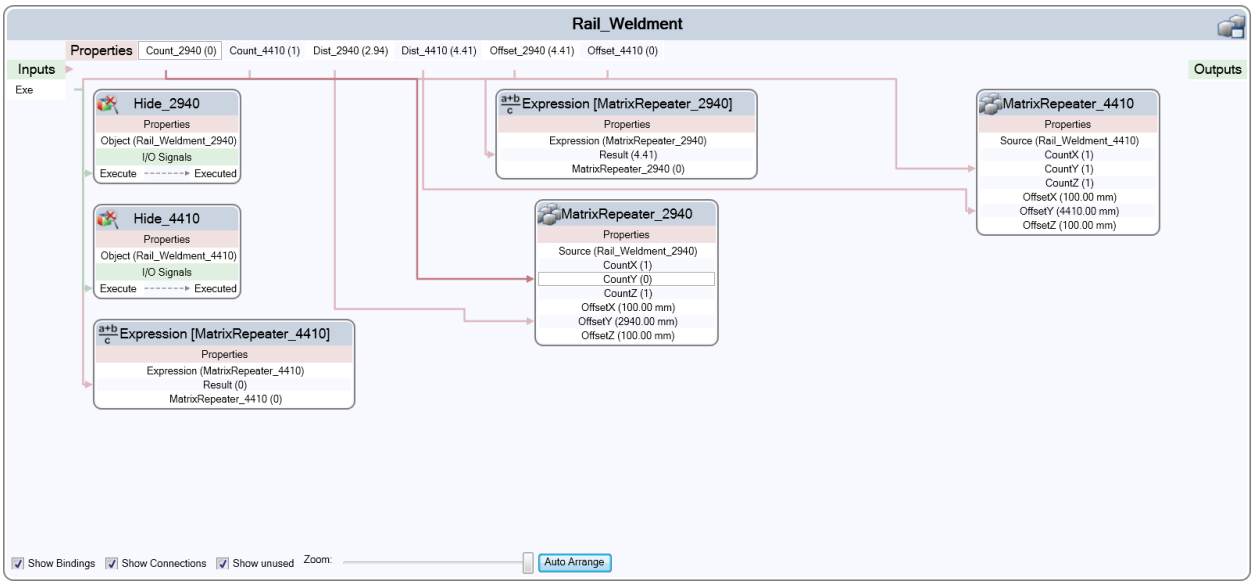


Figure 21: Automatic Track Builder RobotStudio found in Appendix C.

III.1.C – Automatic Light Curtains

The Automatic Light Curtain smart component was programmed dynamically so that multiple light curtains could be imported into the station simultaneously. This component creates a semi-transparent representation of the lasers that spread across the opening of a cell. This can be seen in the figure included below, Figure 22.

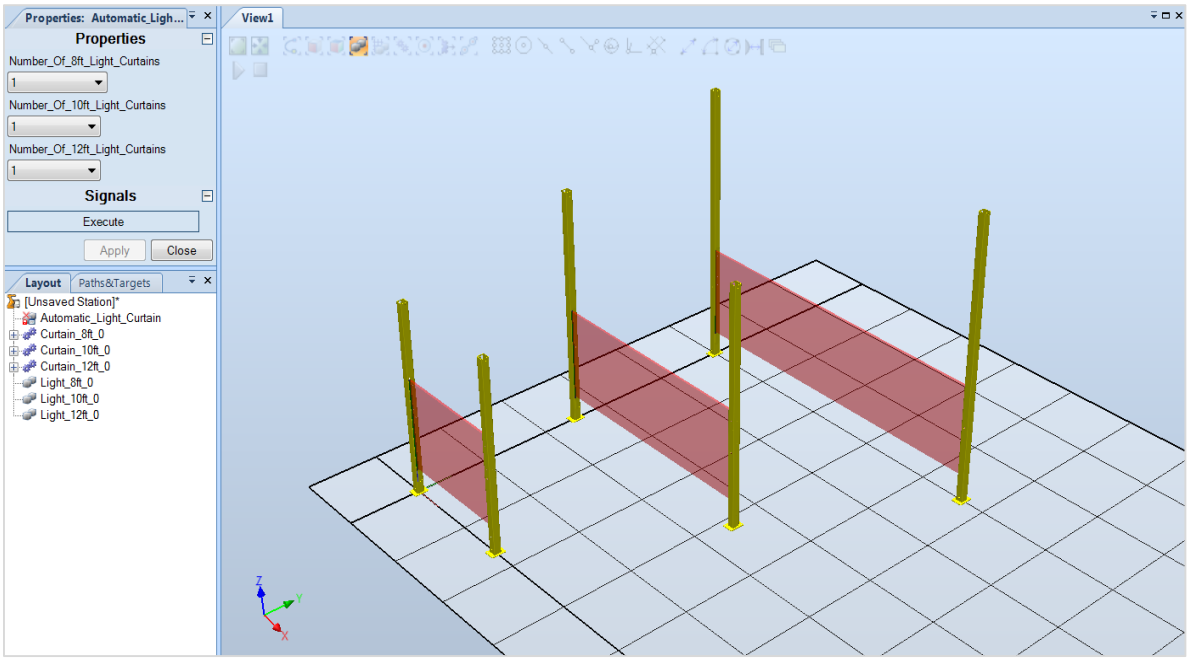


Figure 22: Smart Component Automatic Light Curtains

Three loops drive the program and are based on the number of curtains of each height, seen in Figure 23. This allowed for each individual light curtain to be updated a modified. Initially, a string is declared then edited on each iteration of the loop so component can be called correctly. The code below clears the station of old geometry so new updated geometry can be created.

```
System.Text.StringBuilder sb = new System.Text.StringBuilder("Curtain_8ft_0");
System.Text.StringBuilder sb_light = new System.Text.StringBuilder("Light_8ft_0");

        LOOP:
             sb.Remove(12, 1);
             sb_light.Remove(10,1);
             sb.Append(count);
             sb_light.Append(count);
```

Figure 23: Automatic Light Curtain CodeBehind found in Appendix C.

A preexisting light curtain is searched for and if none are found, one will be created. The joint values, location, and orientation are pulled from the station and used to build the appropriate laser curtain. The code below shows the joint values being used to create the proper length box and the location and orientation of the light curtain being used to set the position of the box. Once all of these variables are

set and the box has been added to the station, the color and transparency of the newly created box is set. This was implemented in the code seen in Figure 24.

```
        //Get Current Joint Values
        double[] CurtainLength;
        CurtainLength = Curtain.GetJointValues();
        Curtain.Frames.ToArray();

        //Create Light Curtain
        Part Light = new Part();
        Light.Name = sb_light.ToString();

        //Move Light Curtain to Correct Location
        Matrix4 rot = Curtain.Transform.GlobalMatrix;
        Vector3 size = new Vector3(CurtainLength[0] + .13096, .03048, 1.20371);

        //Create Body and set Color and Transparency
        Body square = Body.CreateSolidBox(rot, size);
        square.Transform.Z = .33439;
        square.Name = "box";
        Light.Bodies.Add(square);
        VSTABridge.SetColor(Light, 255, 0, 0, 126);
```

Figure 24: Automatic Light Curtain CodeBehind found in Appendix C.

The laser curtain is then attached to the base of the light curtain mechanism so that it changes position with the light curtain if the user decides to move or reorient the light curtain mechanism. If in the initial search a light curtain is found, the associated laser curtain is deleted and rebuilt based on new joint values, location, and orientation of the light curtain. This was implemented in the code seen in Figure 25.

```
//Attach box to base of mechanism
        GraphicComponent home;
        Curtain.GetParentLink(0, out home);
        Part home1 = (Part)home;
        home1.Attach(Light, false, new Matrix4(new Vector3(0, 0, 0)));
```

Figure 25: Automatic Light Curtain CodeBehind found in Appendix C.

The CodeBehind for the Automatic Light Curtain is then compiled and loaded into RobotStudio. The visual programing responsible for this component is much simpler than the components before. This component loads all geometry required and thus only requires input from the user. This input was created in the station and wired into the smart component. This can be seen in Figure 26.

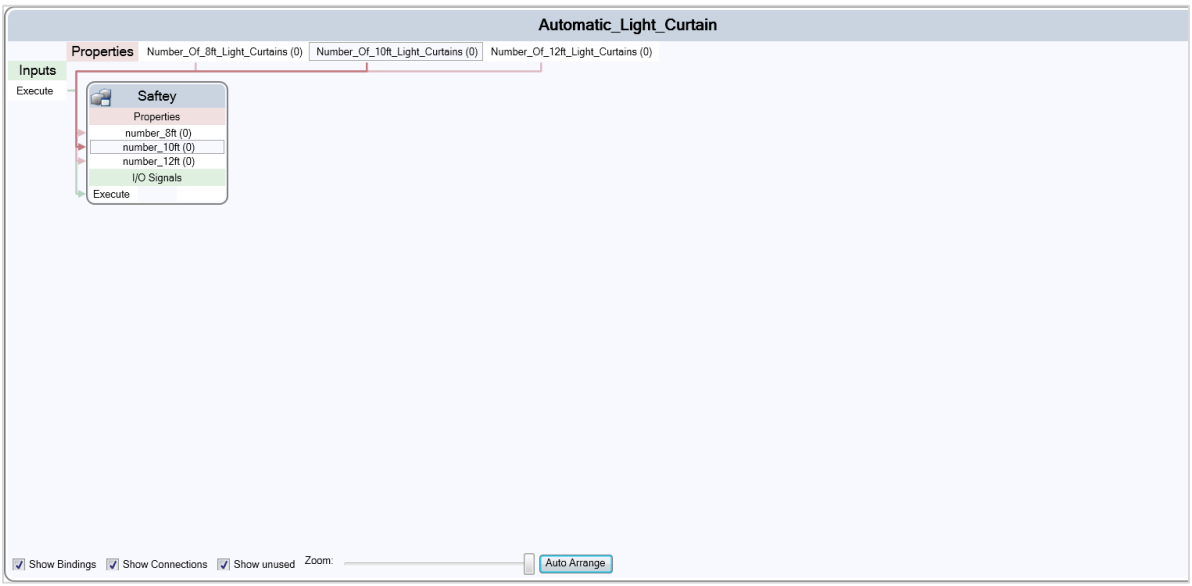


Figure 26: Automatic Light Curtain RobotStudio found in Appendix C.

III.2 – Add-in

III.2.A – Positioners

The positioners add-in was built to be externally deployed for the use of others outside of the company. For this reason, it was built in an easy to use add-in that could be installed on a remote computer with the required external libraries. This code was designed to create a SkyHook positioner, SkyHook positioner with lift, Head Stock Tail Stock combo, Head Stock Tail Stock with lift, or Drop Center using shared geometries to limit the size of the external library needed.

The program begins by creating its own tab and setting that tab to the current view. Inside of the tab it creates its first button *Create SkyHook* and builds the necessary even handler. The image on the button is set to an internal resource called *SkyHook*. The tab and button are then added to the RobotStudio program. The code section seen below, Figure 27, shows how the *SkyHook* button is created and added to the Wolf Positioners tab.

```
//Create a tab
            RibbonTab ribbontab = new RibbonTab("Wolf Positioners", "Wolf Positioners");
            UIEnvironment.RibbonTabs.Add(ribbontab);

            //Make tab active
            UIEnvironment.ActiveRibbonTab = ribbontab;

            //Create a button Group
            RibbonGroup ribbongroup = new RibbonGroup("SkyHook", "SkyHook");

            //Create skyhook button
            CommandBarButton button = new CommandBarButton("Create SkyHook");
            button.Caption = "Create SkyHook";
            button.HelpText = "Create a SkyHook";
            button.Enabled = true;
            button.Image = Properties.Resources.SkyHook;

            //Add button event
            button.ExecuteCommand += new ExecuteCommandEventHandler(button_ExecuteCommand);

            //Add button to ribbon group
            ribbongroup.Controls.Add(button);

            //Add ribbongroup to tab
            ribbontab.Groups.Add(ribbongroup);
```

Figure 27: Positioners Class1 found in Appendix D.

The station then waits for a button depression. When the *SkyHook* button is activated, it builds the form detailed in Form1 and Form1.Designer. This form appears in the user's window and waits for one of its selections to be changed. When a selection is changed, it updates the other selection boxes to reflect the options that are available in the standard Wolf libraries. Figure 28, shows the options for throw comboBox2 and drop comboBox3 if a capacity of 1000Kg has been selected.

```
if (comboBox1.Text == "1000Kg")
            {
              comboBox2.Items.AddRange(new object[] { "400", "500", "600", "700", "800", "900", "1000"
});
              comboBox3.Items.AddRange(new object[] { "400", "500", "600", "700", "800", "900",
"1000", "1100", "1200" });
            }
```

Figure 28: Positioners Form1 found in Appendix D.

When the user is finished with the selection and pushes the *OK* button, the capacity, lift, and throw are stored in a class and the build function is executed. Once inside the build function, the program loads the base, arm, and platter geometries and places them relative to the throw and drop that have been selected. A riser is then selected from the larger of drop and throw. The code segment below, Figure

29, shows a riser being loaded if the drop is larger than the throw. Drop has an additional 800mm of offset to compensate for the motor that turns the platter.

```
if ((Convert.ToInt32(settings.skyhook_drop) + 800) > Convert.ToInt32(settings.skyhook_throw))
                {
                    string riser_path = "S:\\Library Build\\Custom_Lib\\Riser\\WTU5000 Risers\\";
                    mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU5000_" +
Convert.ToString(Convert.ToInt32(settings.skyhook_drop) + 800) + "H.rslib", true);
                    Part Riser = (Part)mylib.RootComponent.CopyInstance();
                    Riser.Name = "Riser";
                    station.GraphicComponents.Add(Riser);
                    Riser.Transform.RY = -90 * (Math.PI / 180);
                    Riser.Transform.RZ = -180 * (Math.PI / 180);
                    Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
                    Riser.Transform.X = (-1811.439 - (Convert.ToInt32(settings.skyhook_throw) - 1500))
* .001;

                    GraphicComponent L1;
                    mech.GetParentLink(0, out L1);
                    Part attacher = (Part)L1;
                    attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));
                }
```

Figure 29: Positioners Class1 found in Appendix D.

Once the appropriate geometry has been loaded and placed, the SkyHook mechanism is built. The code segment below, Figure 30, shows the loaded geometry added to links and joints added to the links. Two vectors dictate where the joint should be placed and what axis it will move about—both of the joints are set to rotational. Once the joint limits have been set, the mechanism is compiled, added to the station, and named.

```
MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
                mb.ModelName = "SkyHook";
                mb.AddLink("Base", Base);
                mb.AddLink("Link1", Arm);
                mb.AddLink("Link2", Platter);
                mb.BaseLink = "Base";
                mb.AddJoint("J1", "Base", "Link1", new Vector3(1, 0,
(Convert.ToInt32(settings.skyhook_drop) * .001)), new Vector3(0, 0,
(Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational, true);
                mb.AddJoint("J2", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
                mb.SetJointLimit("J1", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
                mb.SetJointLimit("J2", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
                Mechanism mech = mb.CompileMechanism();
                station.GraphicComponents.Add(mech);
                mech.Name = "SkyHook";
```

Figure 30: Positioners Class1 found in Appendix D.

III.3 – Automatic Station Builder

The Automatic Station Builder component was built to further leverage the developed smart components. Figure 31, shows the general layout of the Station Builder component. A RobotStudio user interface was developed to support modularity. By creating this overall component, individual smart components could be added or edited without re-wiring the RobotStudio code blocks. This Graphical User interface was wired into the Automatic Station Builder so that important variables could be passed to the Station Builder component and calculations could be performed as necessary. This component acted as the brains for the entire project. This component is responsible for loading the station with the appropriate components, deleting old components, and placing all of the essential components. This component is then wired directly to the subcomponents fencing and Automatic Track Builder. Each of these components has over 40 individual code blocks and wires—not shown.

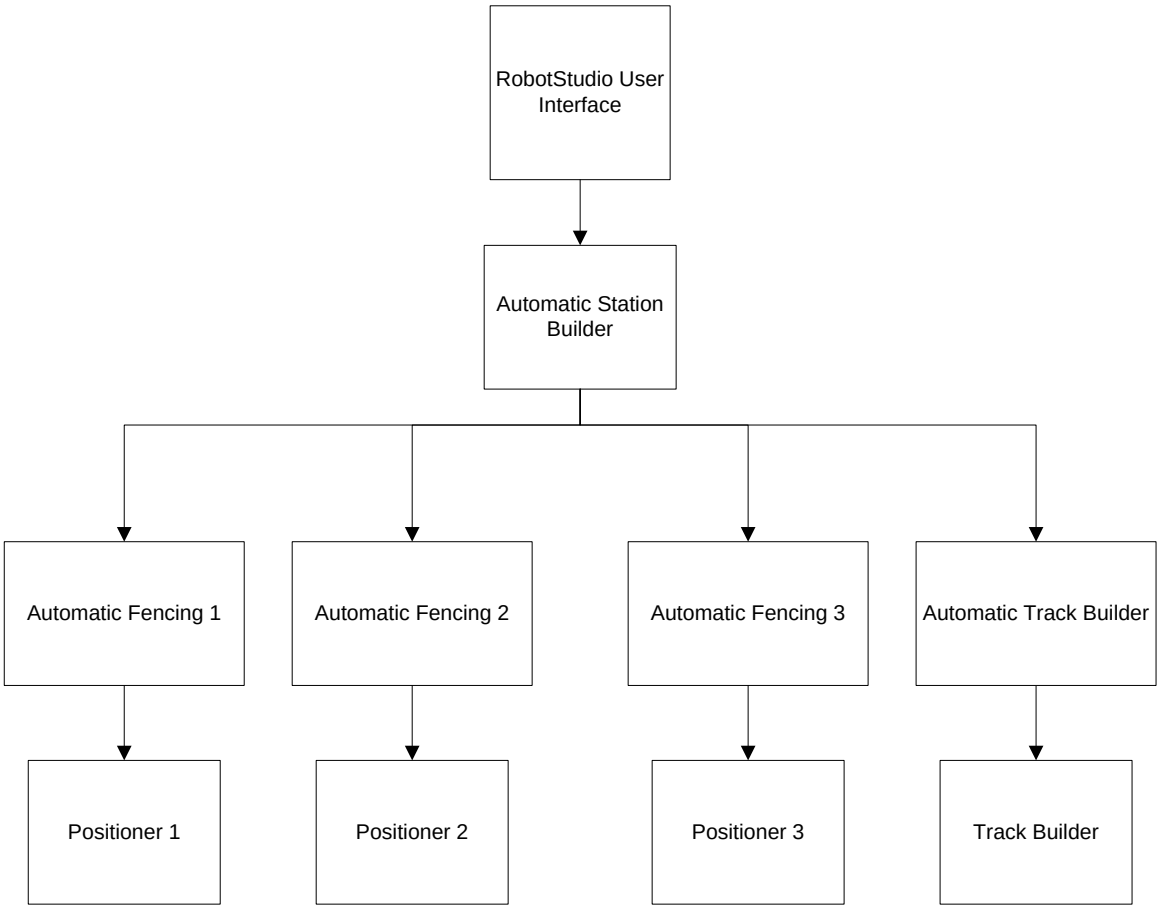


Figure 31: Automatic Station Builder Program Layout

After variables are passed to the Automatic Station Builder calculations are performed to ensure the generated station is capable of welding the desired part. The code segment shown below, Figure 32, shows how the riser and HSTS capacity are selected. It can be seen if a part weight is less than the max capacity of a HSTS combination but greater than the previous HSTS, in this case 0 Kg, that capacity is selected. A list of risers available for the selected HSTS capacity is then iterated through using half of the diameter to determine an appropriate turning center height.

```
//Select HSTS
if (Convert.ToInt32(Orig_Weight) >= 0 & Convert.ToInt32(Orig_Weight) < 3000)
{
   CodeBehind.settings.HSTS_capacity = "3000Kg";

   //Select Riser
   string[] temp = { "0", "1000", "1100", "1200", "1300", "1400", "1500", "1600", "1700", "1800",
"1900", "2000", "2100", "2200", "2300", "2400", "2500" };
    for (int i = 0; i < 15; i++)
    {
        if (Convert.ToInt32(Orig_Diam)/2 >= Convert.ToInt32(temp[i]) & Convert.ToInt32(Orig_Diam)/2 <
Convert.ToInt32(temp[i + 1]))
        {
            CodeBehind.settings.HSTS_riser_height = temp[i + 1];
        }
    }
}
```

Figure 32: Automatic Station Builder CodeBehind found in Appendix E.

Figure 33, shows how the fence height and gantry height are selected from the riser height, which was selected based on the diameter of the desired part. This code shows if the riser brings the turning center of the part to a height less than 2500mm, a 2500 gantry should be loaded.

```
//Select Gantry Height and Fence Height
            if (Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) < 2500)
            {
                component.Properties["Track_Gantry"].Value = ("Gantry_2500H");
                component.Properties["Fence_Height"].Value = (8);
            }
```

Figure 33: Automatic Station Builder CodeBehind found in Appendix E.

Further calculations to determine the length of the fence and placement of each component can be seen in Figure 34. This code shows the offsets used for a 20,000Kg HSTS combination. The Math.Ceiling Function was used here to account for rounding errors in the different smart components.

```
if (settings.HSTS_capacity == "20000Kg")
{
     component.Properties["Fence_Length1"].Value = Math.Ceiling((4000 + CodeBehind.settings.RTT) *
3.28084 * .001);
     component.Properties["Fence_Length2"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4000) * .001));
     component.Properties["Fence_Length3"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4000) * .001));

     component.Properties["Fence_Pos3"].Value = (new Vector3((-2000 * .001), (Math.Ceiling((4000 +
CodeBehind.settings.RTT) * 3.28084 * .001) * 309.48 -2951.24) * .001, 0));

     CodeBehind.settings.Curtain_Transform_Y = -3095.887 * .001;
     CodeBehind.settings.Curtain_Transform_X = (Math.Ceiling(3.28084 * .001 *
Convert.ToInt32(CodeBehind.settings.HSTS_riser_height)) * 309.48 + 1961.08 + 295.44) * .001;
     CodeBehind.settings.Curtain_Len = (Math.Ceiling((4000 + CodeBehind.settings.RTT) * 3.28084 *
.001) * 309.48 - 2160.69);
}
```

Figure 34: Automatic Station Builder CodeBehind found in Appendix E.

The root equation used in the fence offsets and light curtain offset was determined experimentally. Depending on the length of fence desired a different number of fence segments are required. The additional fence segments also create an additional offset from the fence posts added. Various fence lengths were loaded and the center of the last post from the starting corner was measured. These results were graphed in Figure 35, to determine if a simple relationship could be found. Fortunately, a perfectly linear relationship was found, noted by an R^2 of 1. Additional offsets were then required to compensate for the different sizes of positioners and their placements.

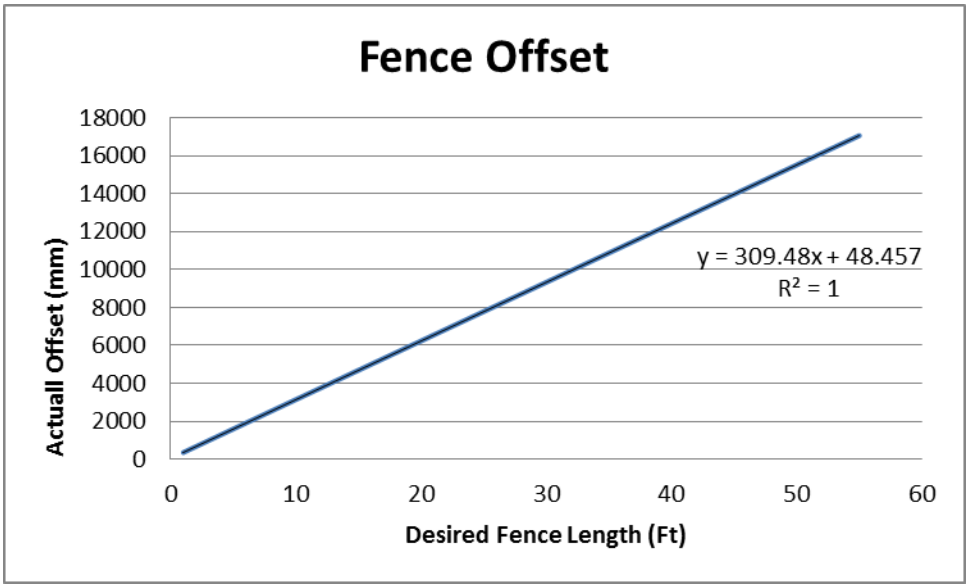


Figure 35: Fence Offset Calculation

IV – CASE STUDY

This section will present a case study of the generation of a robotic concept station in ABB's RobotStudio. The first subsection will perform the development of a robotic concept station without the use of the add-in and smart components. The smart component series and add-ins will then be utilized to build the same station, found in the second subsection. This will show the difficulties that previously existed in the development of concept stations. It will also accent the ease of using the smart component series.

IV.1 – Without Using Add-in or Smart Component Series

In the Home tab of RobotStudio, select *Import Library* then *Browse For Library*. This can be seen in Figure 36. Browse to the folder that contains the Head Stock prebuilt libraries. Select a *30000Kg Head Stock* and select *Load*.

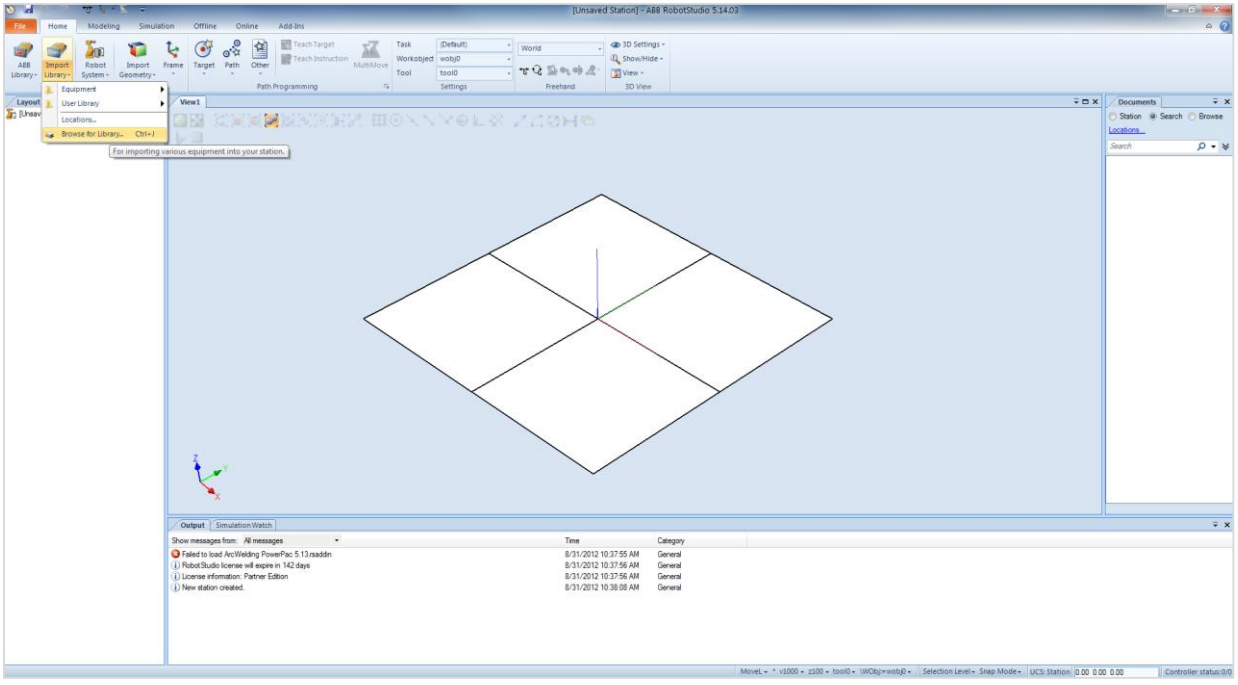


Figure 36: RobotStudio Import Library

Some repositioning may be required in order to see the Head Stock. This can be accomplished using the move tool or repositioning the Head Stock using the coordinate system, seen in Figure 37. It is

important to move it away from the world coordinates (0,0,0) as this is where the robot travel track will be loaded.

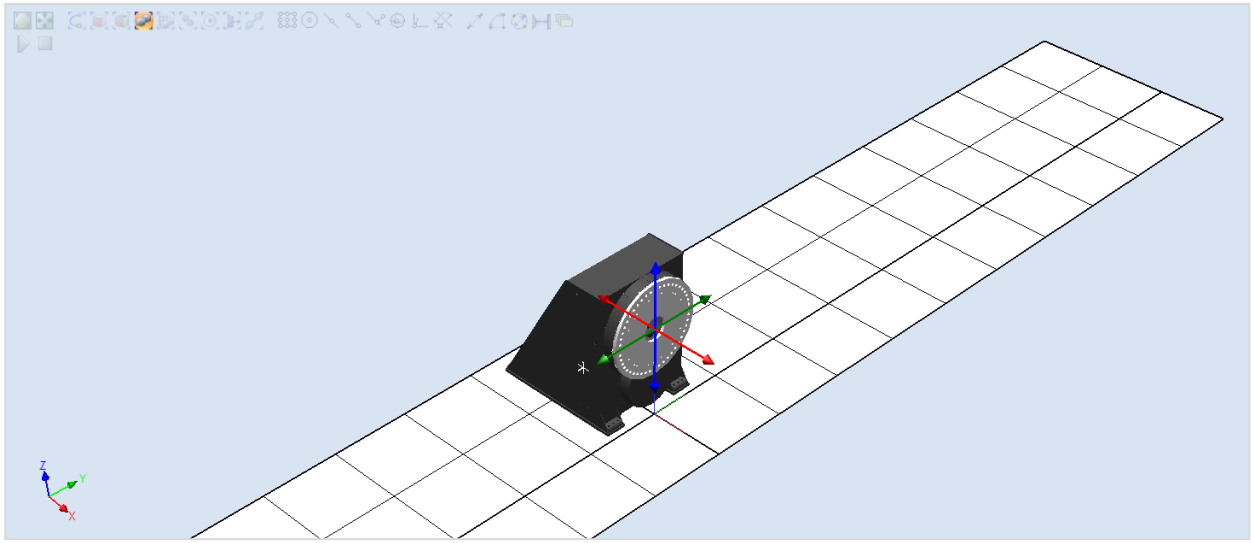


Figure 37: RobotStudio Position Head Stock

Again, use the *Import Library* button to load in the matching Tail Stock. Reorient and position the Tail Stock across from the Head Stock, an example of this can be found in Figure 38. At this point it is not important to position the Tail Stock exactly opposite the Head Stock.

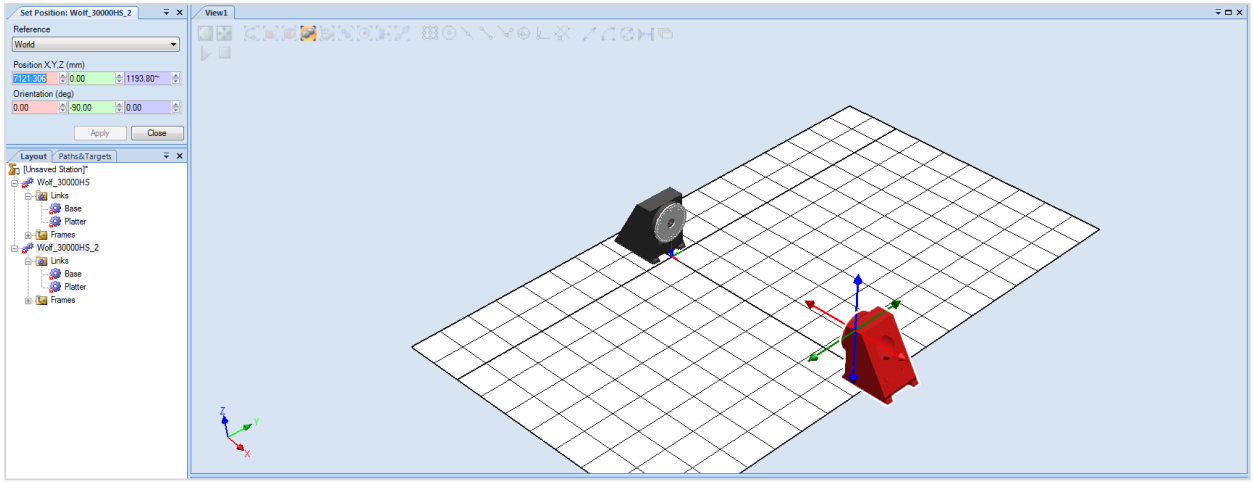


Figure 38: RobotStudio Position Tail Stock

Use Import Library to load two 2500mm risers for a 30,000kKg Head Stock and a 30,000Kg Tail Stock. Reorient the risers so the bolt holes are at the top and position it on the floor. Move the Head Stock to

a position such that the bolt holes in the bottom can be seen, seen in Figure 39. Use the *place one point* function and select one of the bolt holes. For the second point, select the matching bolt hole on the riser. This will move the Head Stock to the proper position on the riser.

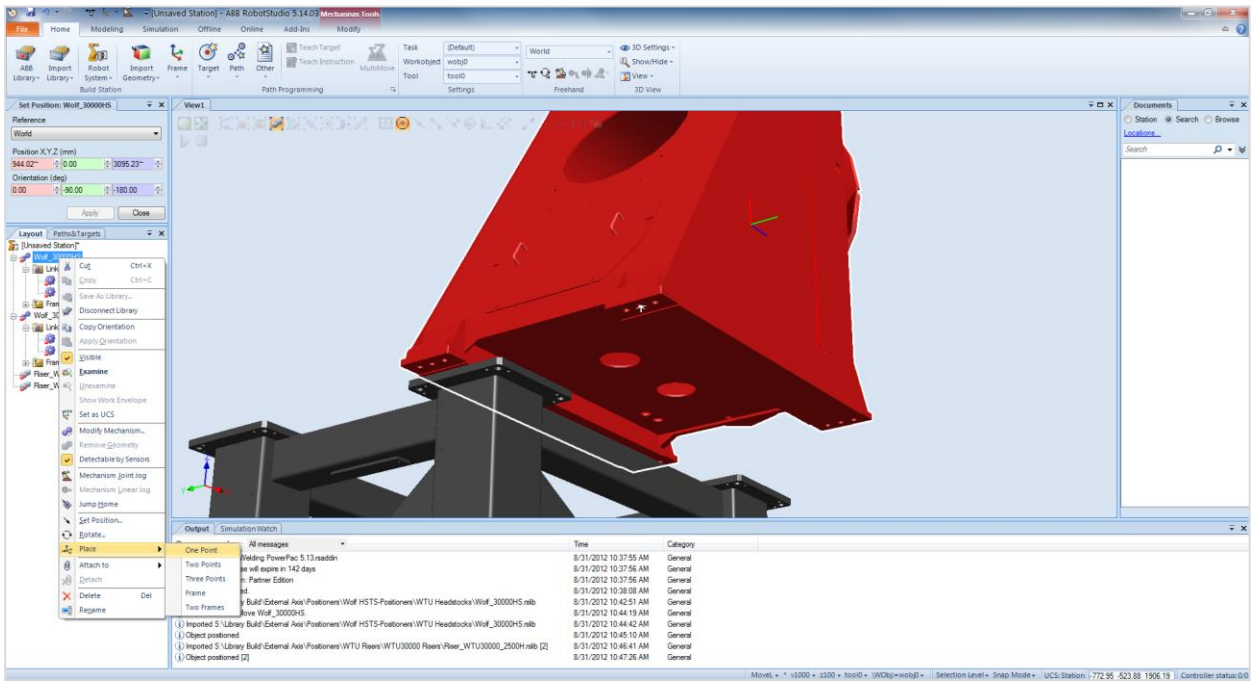


Figure 39: RobotStudio Place Head Stock on Riser

Now attach the Head Stock to its riser. This will allow a user to move the components together. This should look similar to the example seen in Figure 40. Use this procedure again to attach the Tails Stock to its riser.

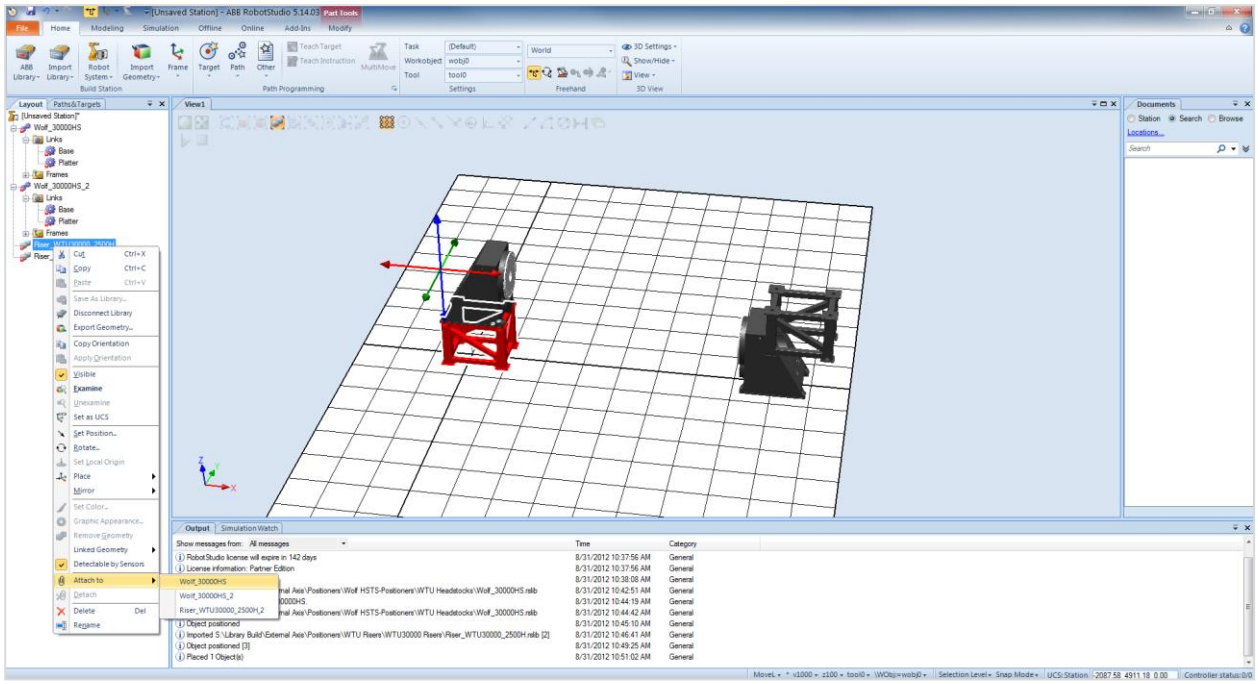


Figure 40: Attach Head Stock to Riser

Load the heavy duty track with a length of 7350mm. Position the track to World Coordinates (0,0,0).

Load a 3500mm Gantry to the station and place it at World Coordinates (0,0,0). When loaded this should have a gantry and a track of 7350mm, seen in Figure 41.

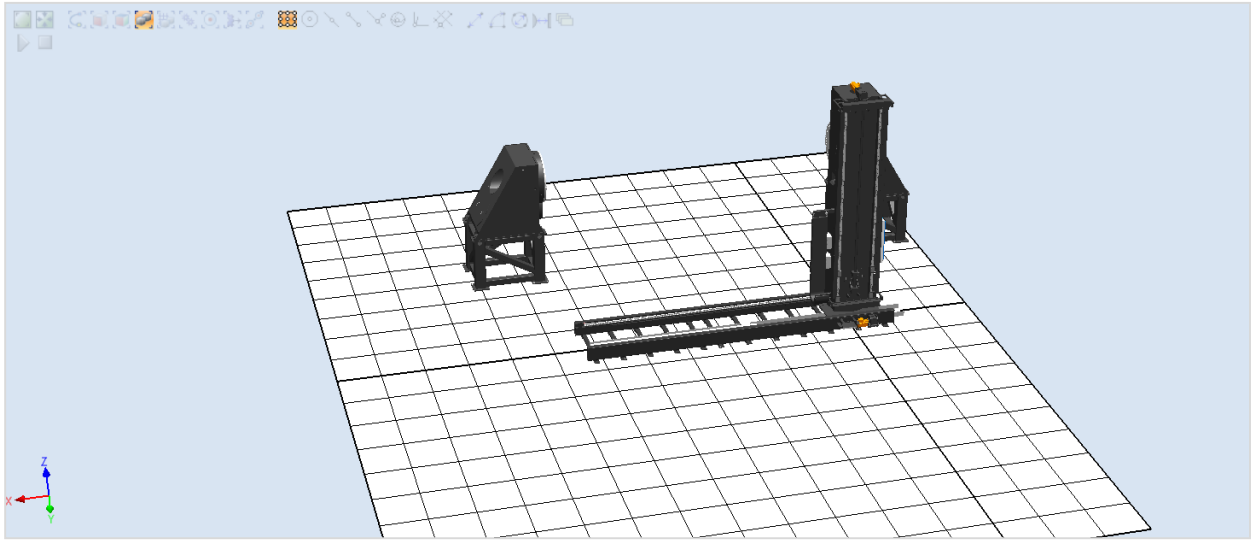


Figure 41: RobotStudio Load Robot Travel Track

Load a 2500 mm boom and orient it to the correct rotational position to match the gantry. Again, use the *place one point* function and the bolt holes to place the boom. Attach the boom to Link 4 of the Gantry. When the station asks if its current position should be kept, select *yes*. An attached boom can be seen in Figure 42.

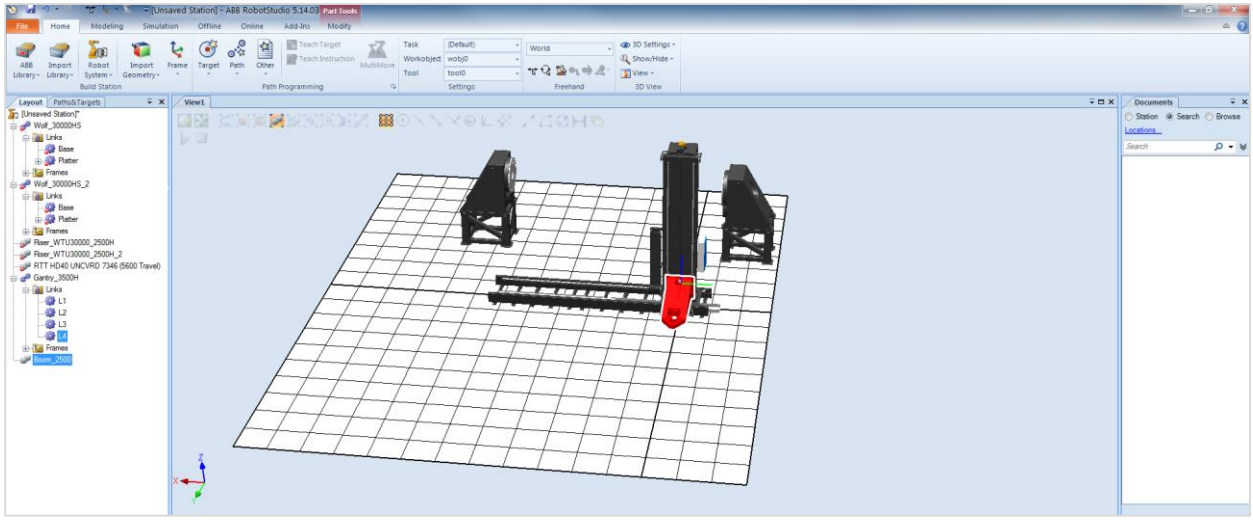


Figure 42: RobotStudio Load Boom

Notice that when using the mechanism joint jog, it is possible to move the gantry all the way off the track, seen in Figure 43. This boom is set up for any length of track.

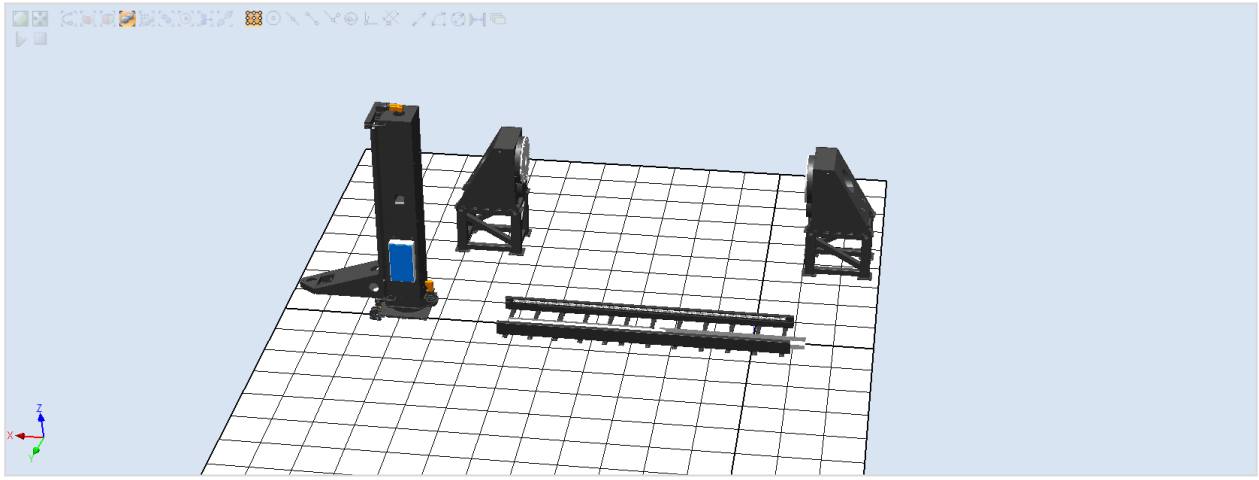


Figure 43: RobotStudio Modifying Gantry Mechanism

Load a Lincoln water cooler, a single wire barrel with a low mount, a torch cleaner, and a Wolf bullseye. Move these parts so all of them are in view. An example of this can be seen in Figure 44.

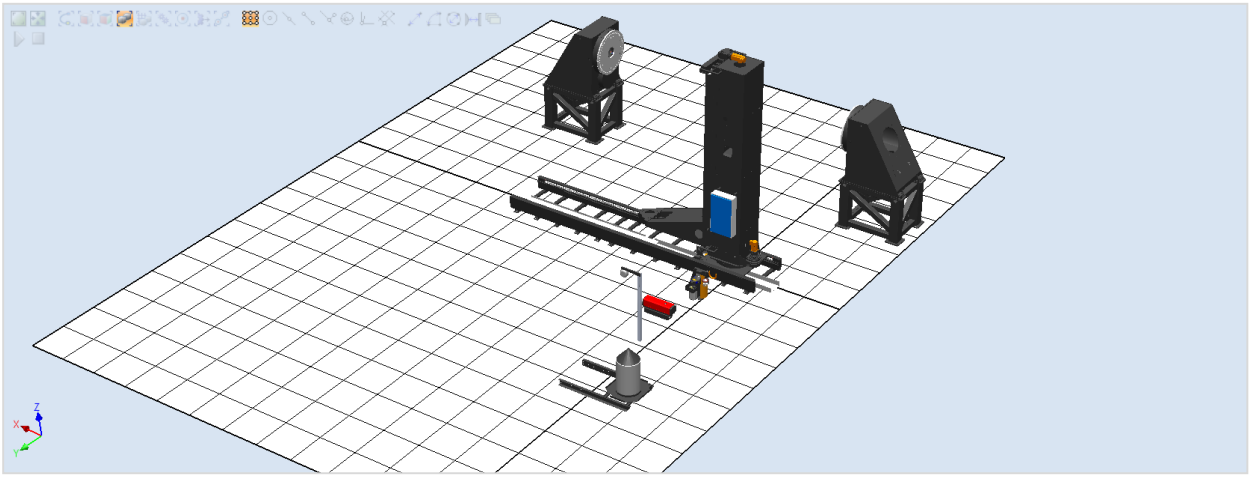


Figure 44: RobotStudio Load Additional Tooling

Individually move each part to its appropriate place on the gantry. Once in position, attach it to the appropriate form. For the Wolf Bullseye and wire management, select link four. For the wire barrel and water cooler, select link three. The final assembly should look similar to the one seen in Figure 45.

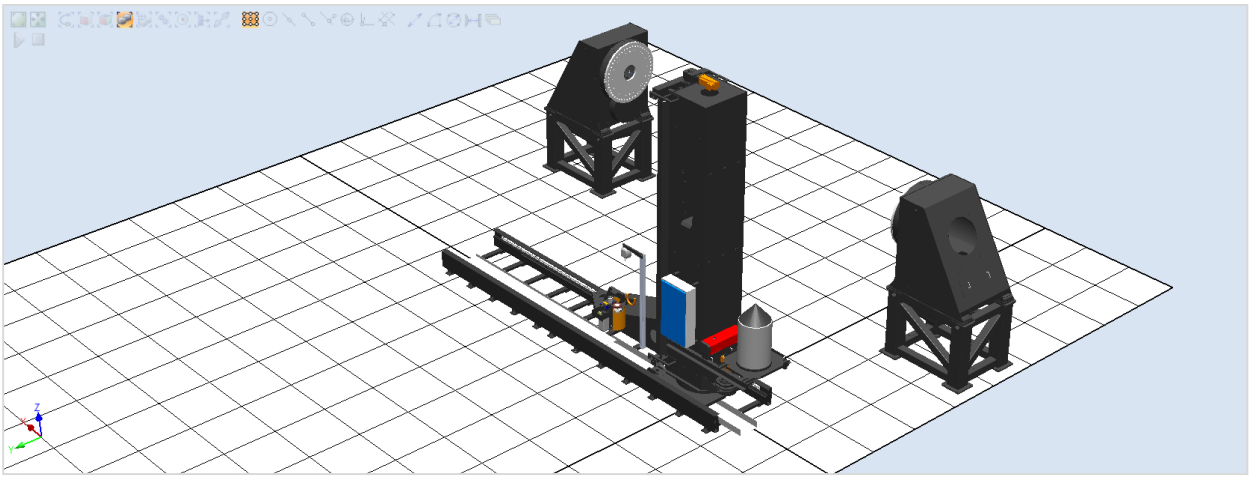


Figure 45: RobotStudio Attach Tooling

Now the fencing must be loaded. Eight sections of 10’x40’ fence and two sections of 10’x5’ must be loaded. Rotate each piece until there are four sections along the x axis and six sections along the y axis. Two loaded fence pieces can be seen in Figure 46.

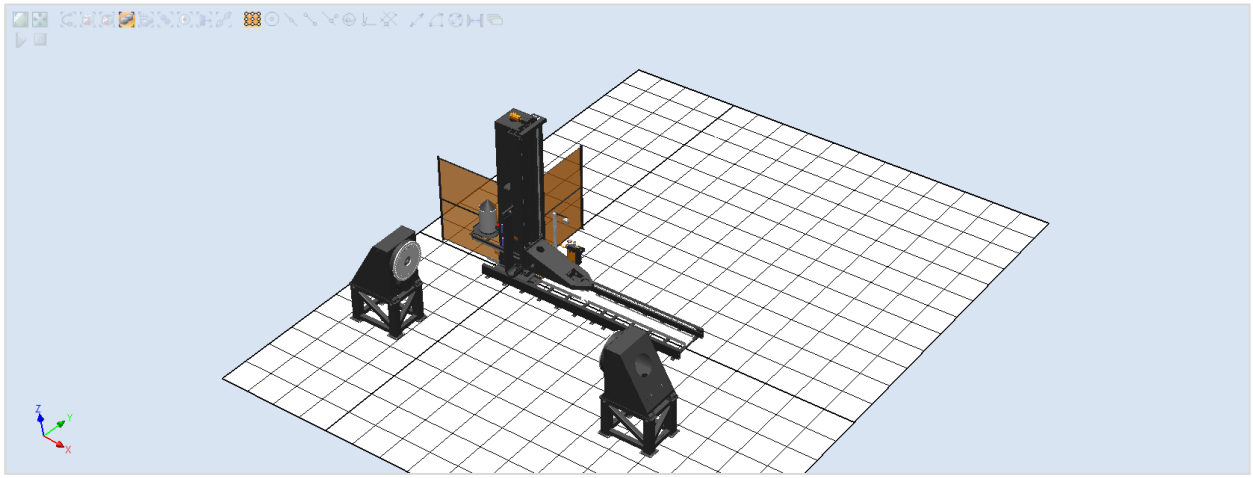


Figure 46: RobotStudio Load Fencing

Each piece of fence must be individually placed. This can be done accurately by either using the coordinate system or by using the move tool and matching up each fence post. The error over many sections of fence will be compounded, so it is best if the coordinate system is used for placement. Figure 47, shows two fence pieces that are miss-aligned.

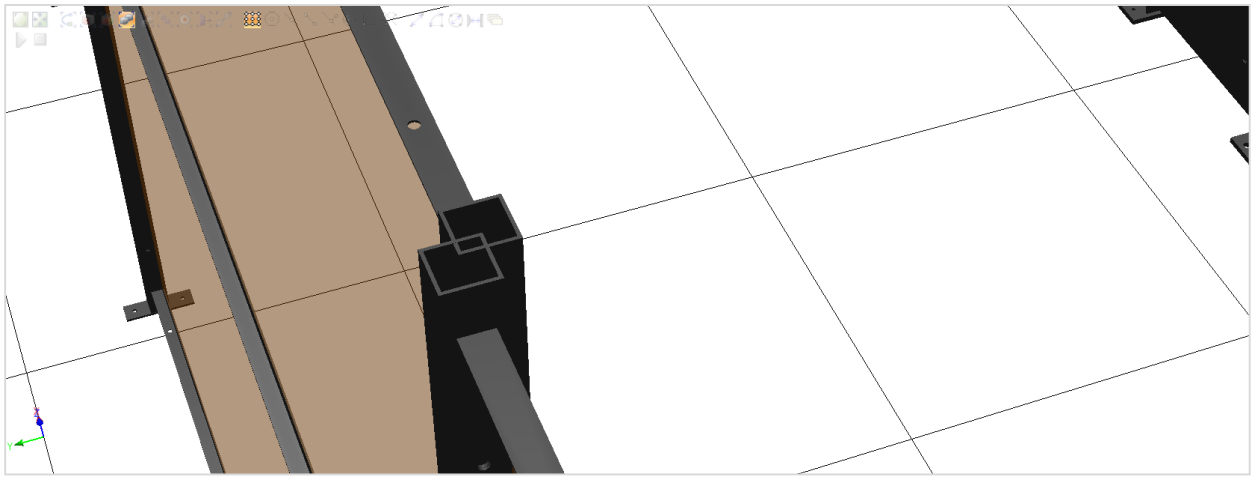


Figure 47: RobotStudio Align Fencing

It can now be seen that not enough fencing was set up, Figure 48. In order to adjust for the shortage, some fencing should be individually deleted and longer fence pieces should be brought in. Again orient and place them in the desired position. If the height of the fence must be changed, each piece of fence must be deleted and new fencing of the proper height needs to be brought in. Again each piece must be individually oriented and placed.

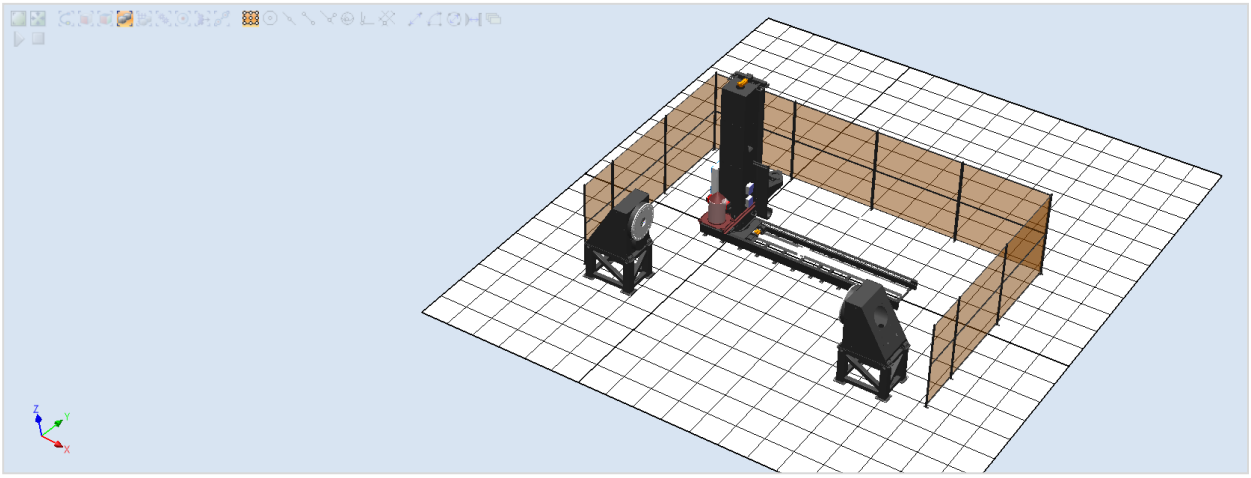


Figure 48: RobotStudio Load Additional Fencing

Once the correct fence segments have been placed, bring in a light curtain pole. Orient and place the pole so that both light curtains are opposite each other and on the corresponding fence post. Any error in the fence placement can easily be seen when the light curtains do not match up. Properly placed fencing and light curtains can be seen in Figure 49.

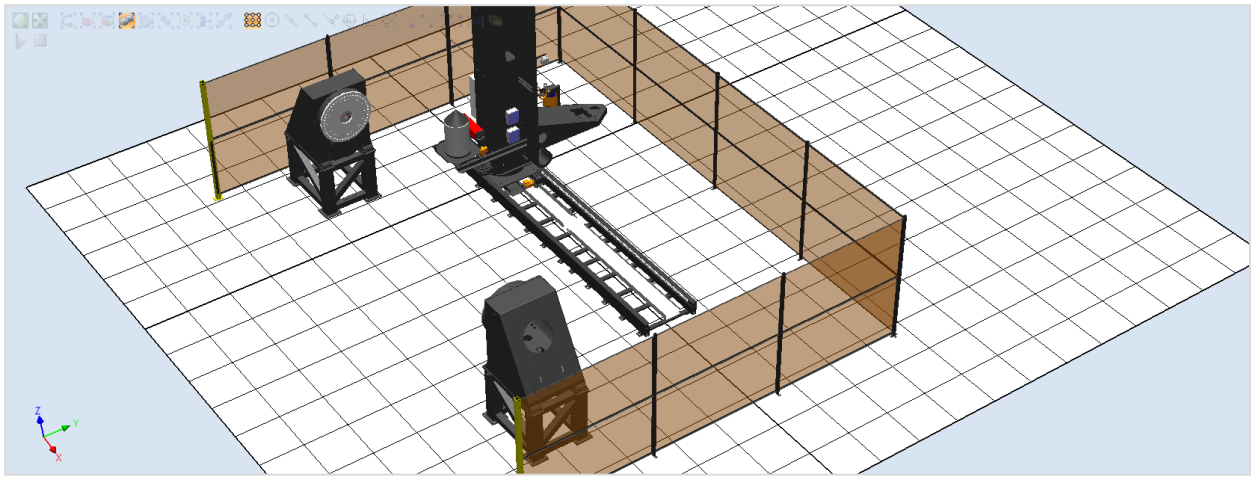


Figure 49: RobotStudio Load and Place Light Curtain

The next step is to build the curtain box. In the modeling, select *solid* then *box from three points*—seen in Figure 50. Select the top right corner of the light curtain then the bottom right corner of the opposite fence post. Again, select the bottom right corner to determine the gap the fence curtain will span.

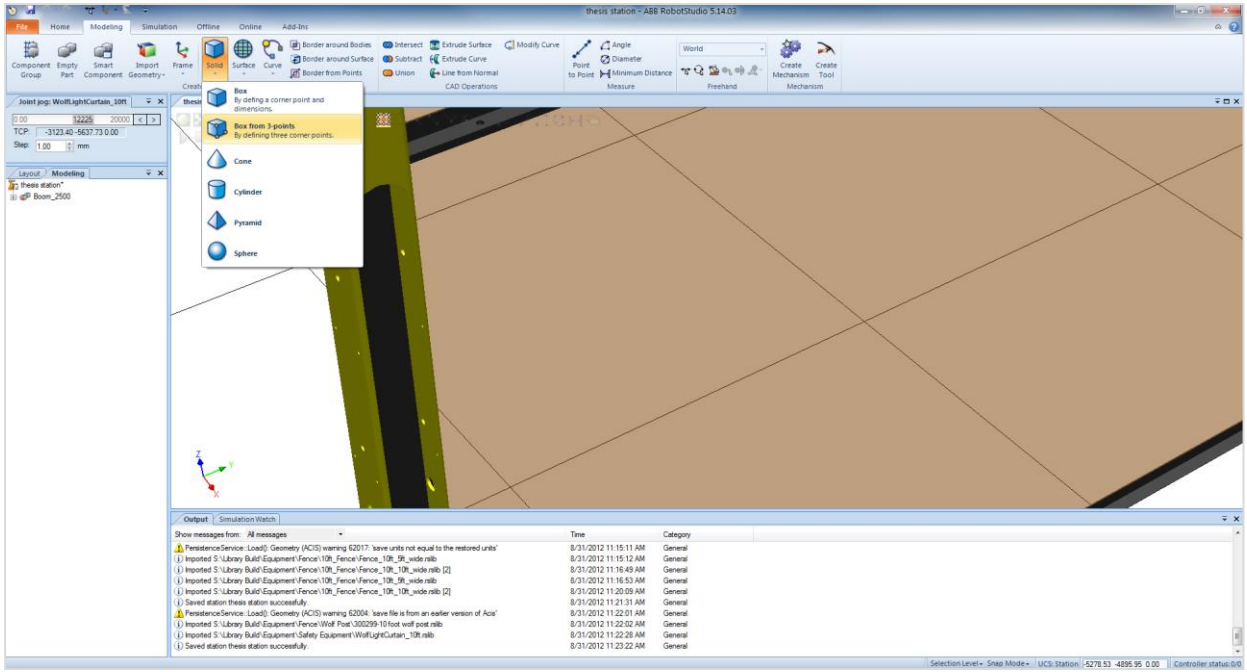


Figure 50: RobotStudio Building the Curtain Box

Select the newly created box and the modify tab will appear; select the newly created box and choose *graphic appearance*. In the dialog box, change the color to red and the transparency to 50%. The station with completed laser curtain can be seen in Figure 51.

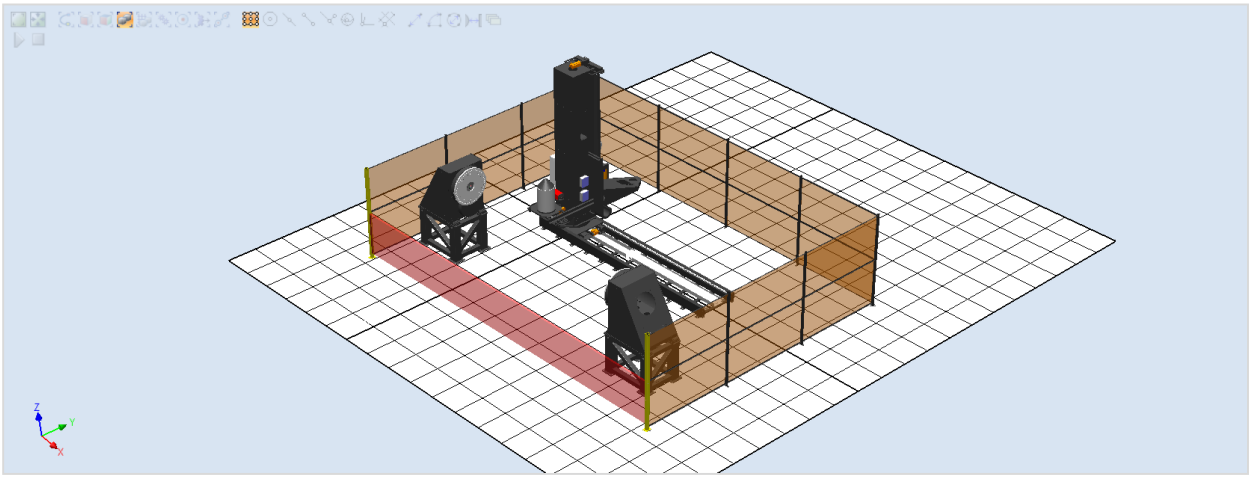


Figure 51: RobotStudio Laser Curtain Complete

A robot can now be selected and brought into the station. Use the move tool to position the robot on the boom. Attach the robot to link four of the gantry. The ready to use concept station can now be seen in Figure 52.

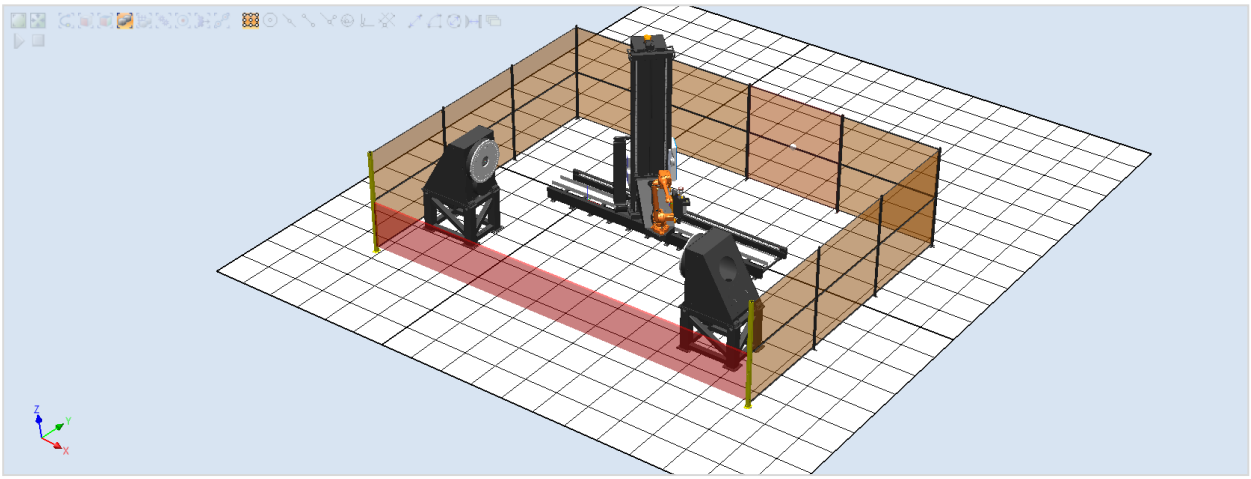


Figure 52: RobotStudio Final Concept Cell

IV.2 – Using Add-in and Smart Component Series

This case study will cover the basic usage of the add-in and smart component series. Begin by creating a new RobotStudio station. Move to the *Wolf Positioners* tab and click the *Create HSTS* button. In the new window that opens, select a capacity of 30,000Kg and riser height of 2500mm. Click the load button. An example dialog box can be seen in Figure 53.

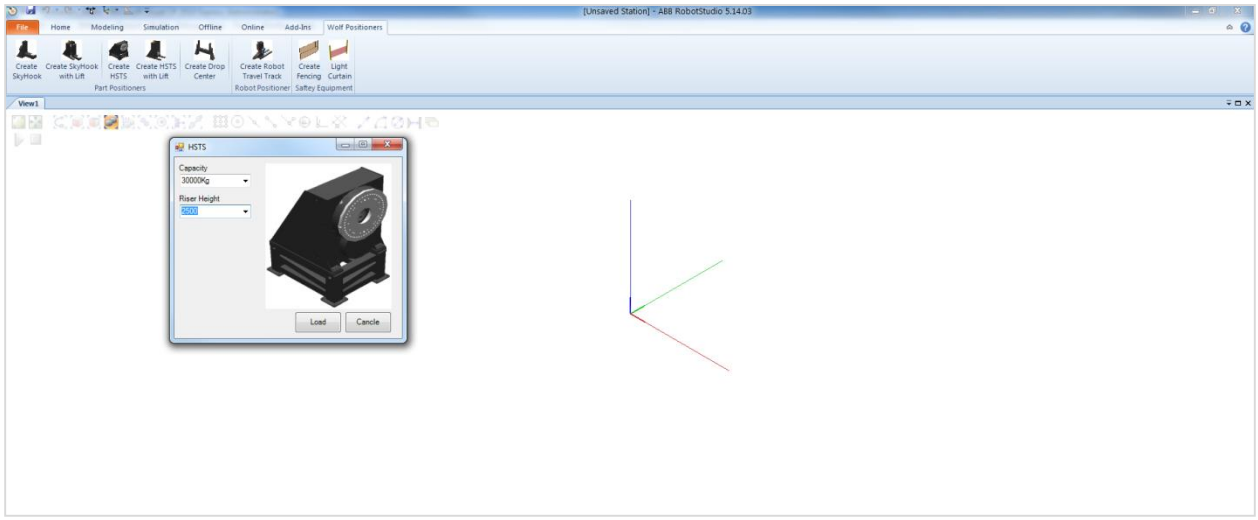


Figure 53: Add-in Load Riser

Now that a positioner is in the station, add the smart component series. This includes automatic track, automatic light curtain, and three automatic fencing. Select the automatic track component; for track length select 7350, for Gantry select the Gantry_3500H, for Boom select Boom_2500. Also, have this component automatically bring in a Lincoln water cooler, a single wire barrel with a low mount, a torch cleaner, and Wolf bullseye. Select *apply* to bring in the new changes. This dialog box can be seen in Figure 54.

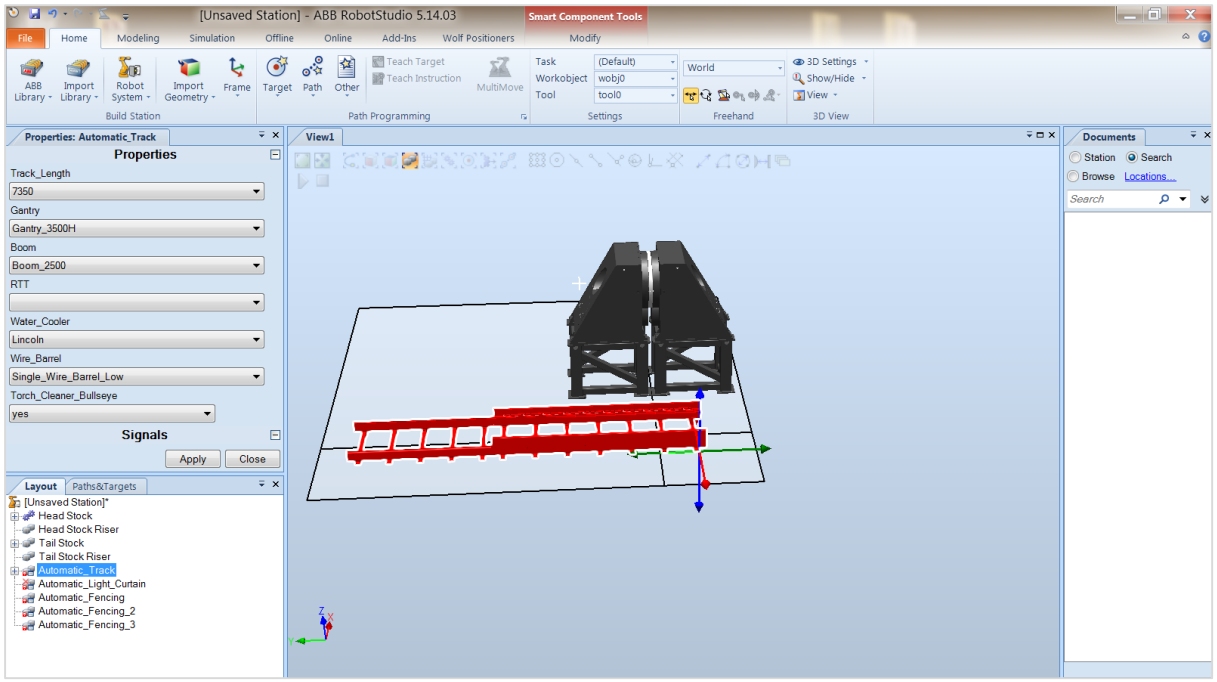


Figure 54: Add-in Load Robot Travel Track

Now move the positioners to the end of the travel track. With the three Automatic Fencing components create two 10’ x 40’ sections and one 10’ x 40’ section. Position the 40’ fence along the back side of the station. Position the two 35’ sections along the sides of the station. This can be done by entering the length in feet and selecting 10 from the Height selection box. Click the *apply* button to complete the changes. After all these changes have been applied the station should look as seen in Figure 55.

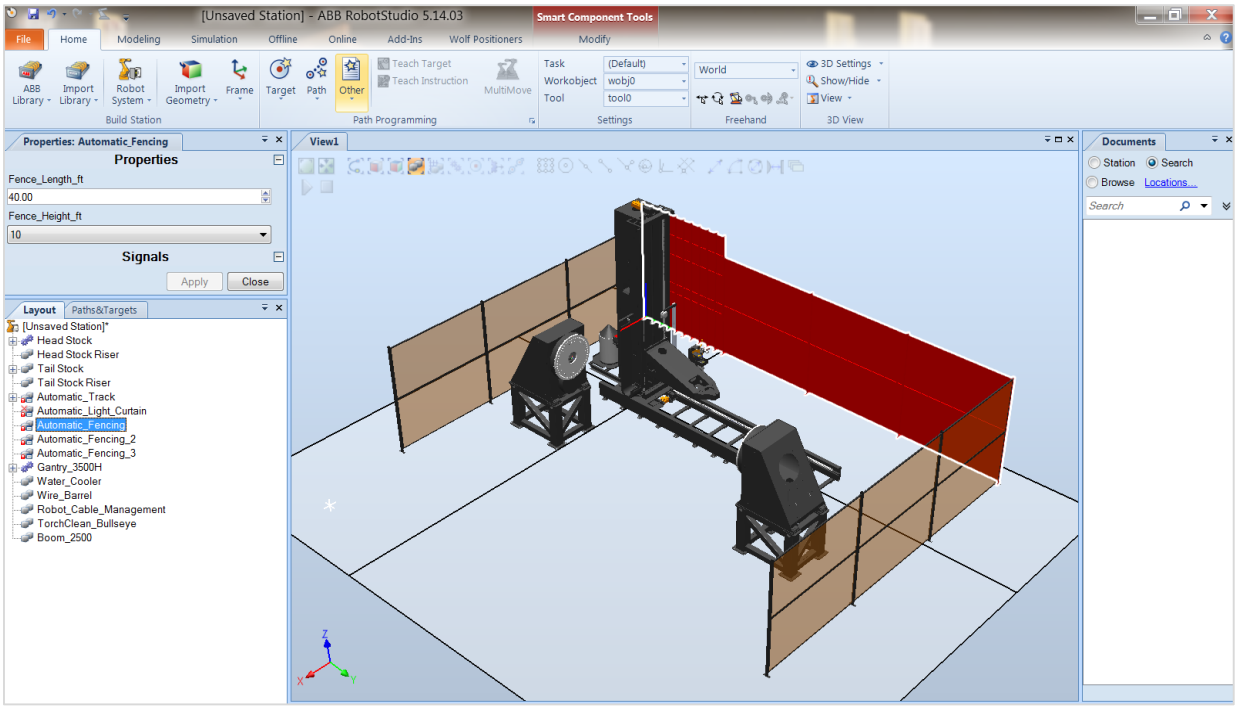


Figure 55: Add-in Load Fencing

Select the *Automatic Light Curtain* component. Select one 10ft Light curtain and select *Apply* then *Execute*. Move the light curtain from world zero out to where it can be seen. Place one post of the light curtain on the last post of the fence. Use mechanism joint jog to move the other light curtain post so that it is on top of the opposite fence post. The light curtain should now be positioned similar to the one seen if Figure 56.

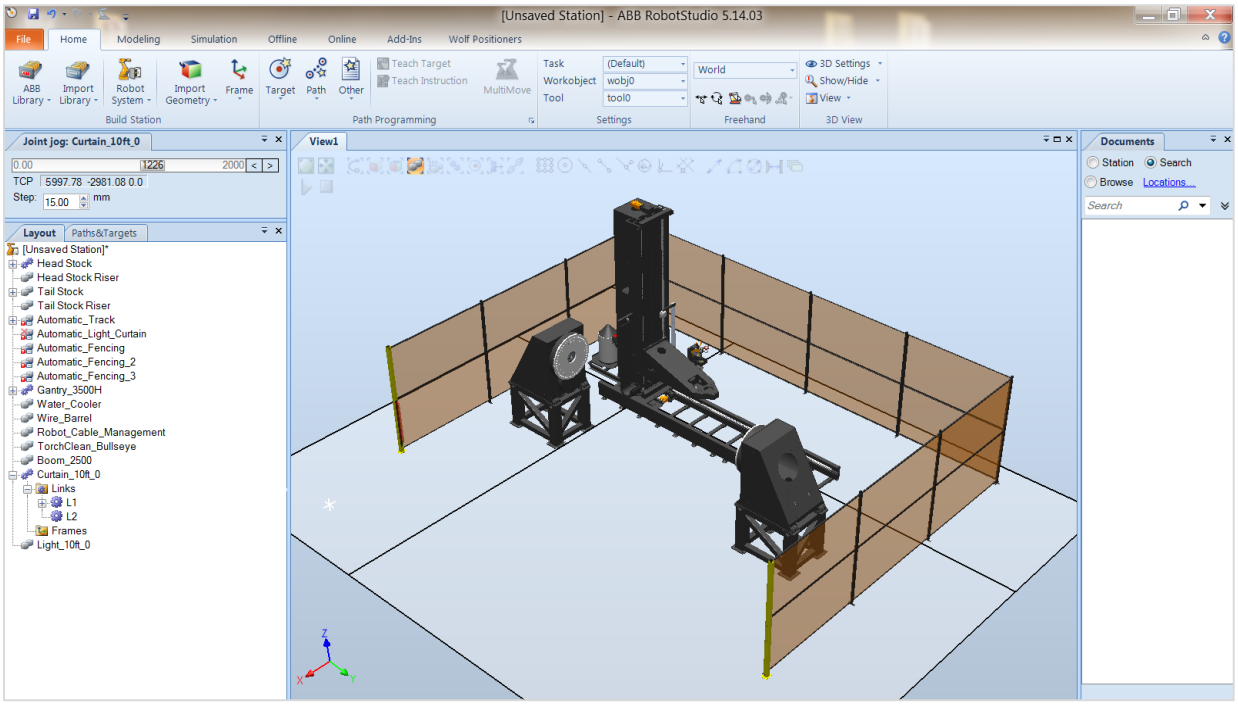


Figure 56: Add-in Load Light Curtain

Go back to the Automatic Light Curtain component and select *Execute* to update the light curtain. This will create a visual representation of the laser that covers the cell opening, seen in Figure 57.

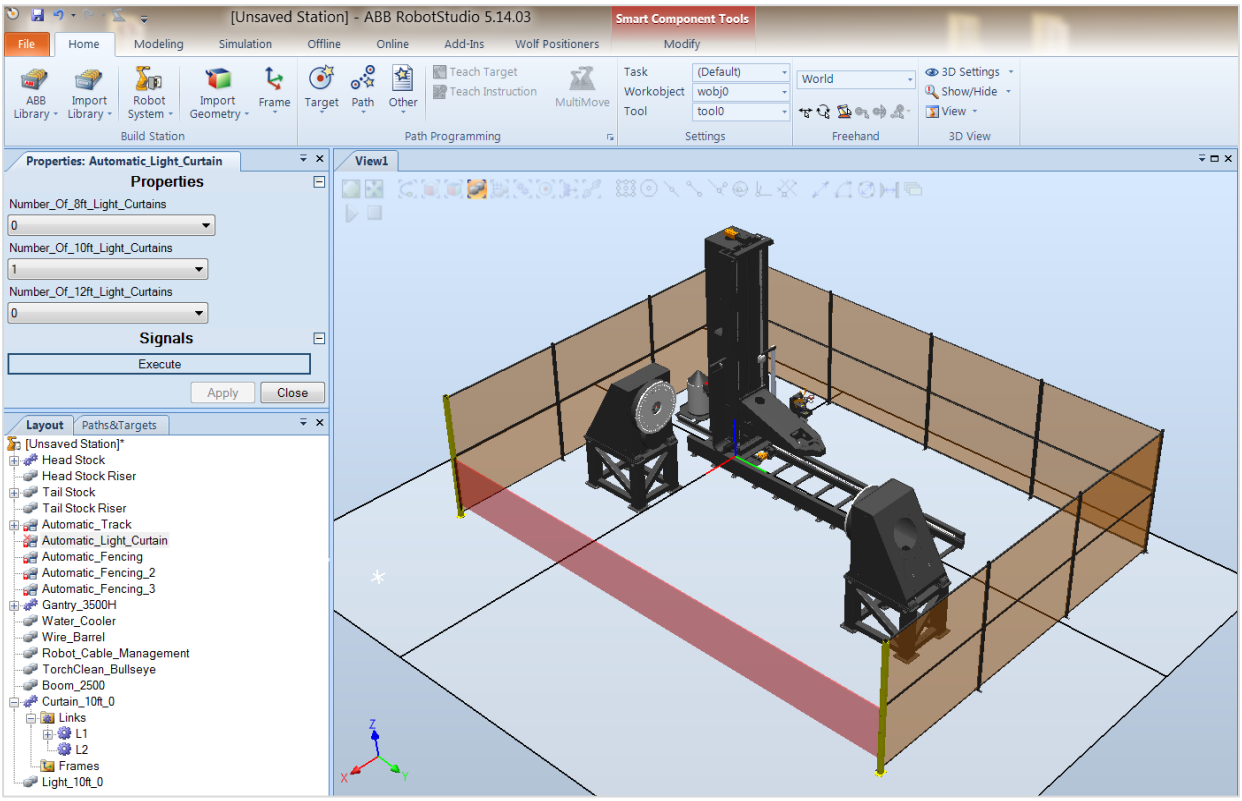


Figure 57: Add-in Update Laser Curtain

Finally bring in a robot and attach it to the gantry. The station is now ready to bring in the customers part and attach it to the Head Stock. Once this has been completed a controller that matches the station mechanisms can be loaded. The station is now ready for programming. The completed station can now be seen in Figure 58.

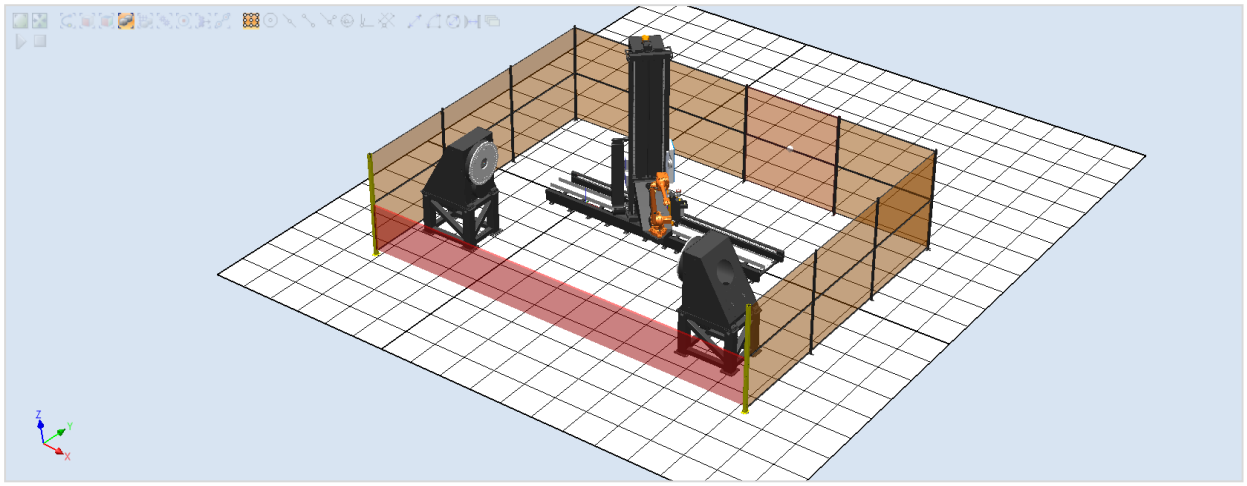


Figure 58: Add-in Final Concept Cell

IV.3 – Using Automatic Station Builder

After loading the Automatic Station Builder component, by double-clicking the icon in your file browser, the screen seen in Figure 59 will load. On the left-hand side of the screen a dialog appears to insert the weight in Kg, diameter in mm, and the length of a part in mm. Pressing *apply* will save these selections. After saving the selections press the *Execute* button.

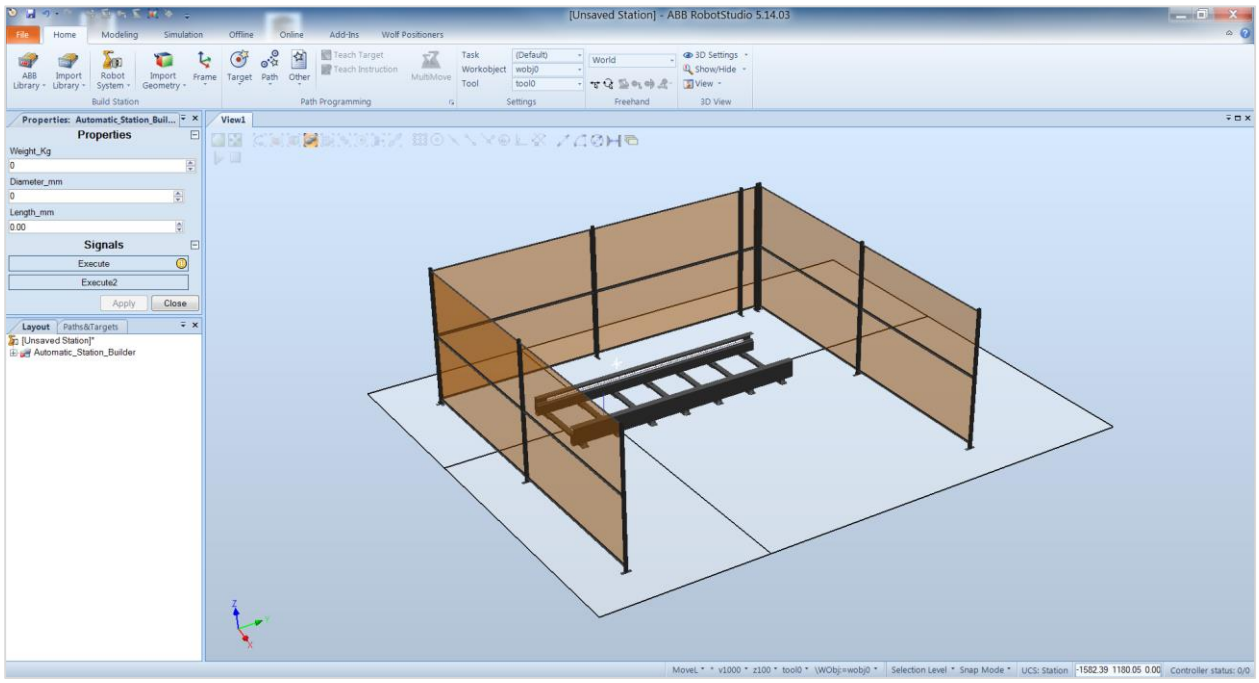


Figure 59: Automatic Station Builder Interface

Pressing the *Execute* button activates the first stage of the component and the station will populate with the components that best match constraints of your part. Figure 60 shows an example of pressing the Execute button with a weight of 50kg, Diameter of 1550mm, and Length of 5580mm.

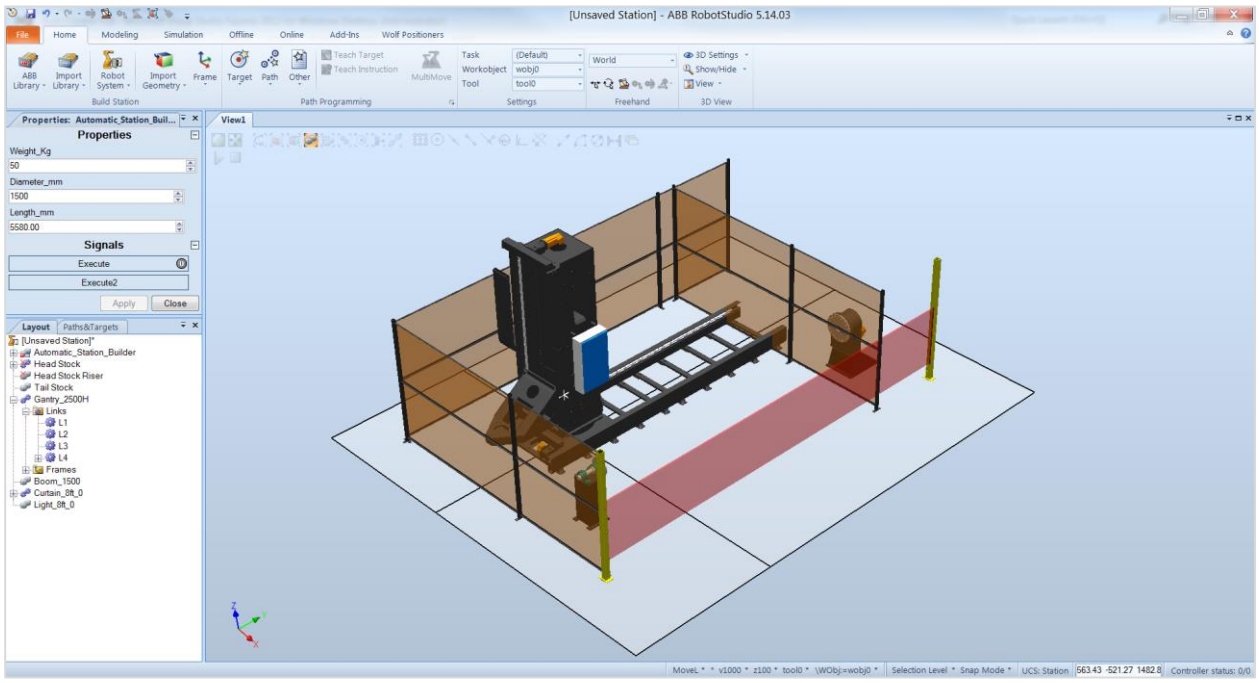


Figure 60: Automatic Station Builder Execute

Now press the *Execute2* button. This button will appropriately place the rest of the components in the generated robotic welding cell. Figure 61, shows the same cell after the components have been placed.

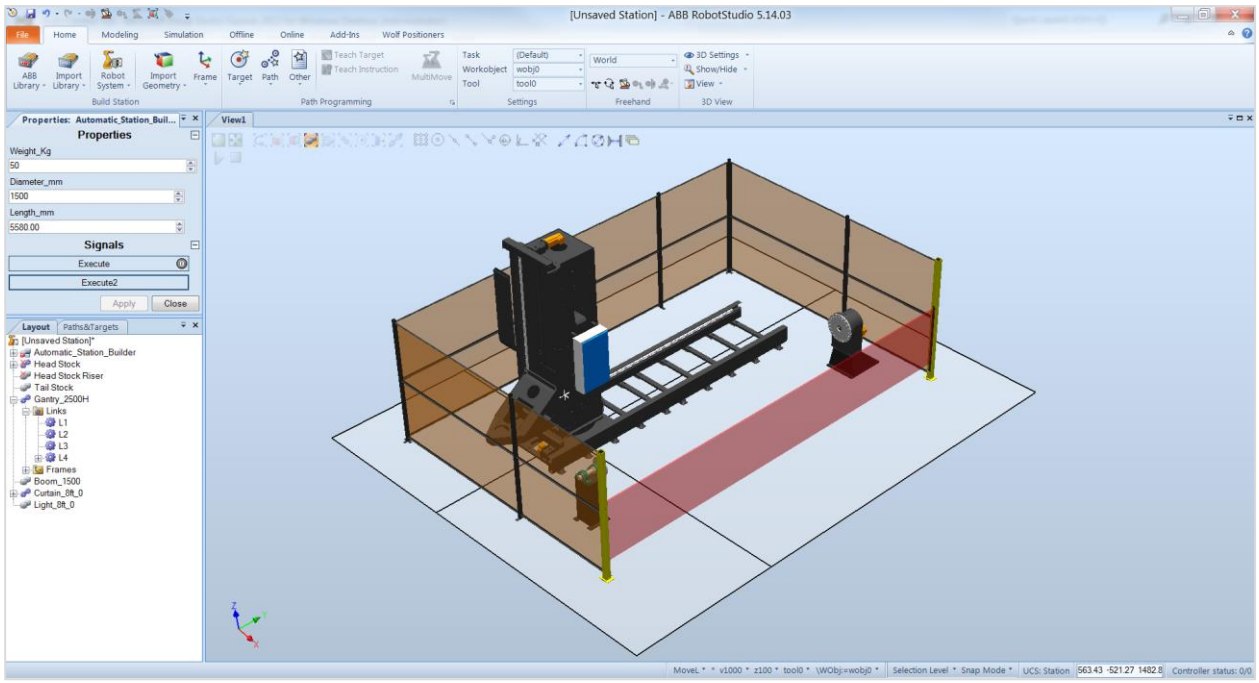


Figure 61: Automatic Station Builder Execute 2

After the robotic welding cell has been constructed a user can now place the part to be welded. The station constructed in Figure 62 now has a robot placed on the robot positioner, and the part with the before described attributes. At this stage the robotic cell weld paths can now be programed.

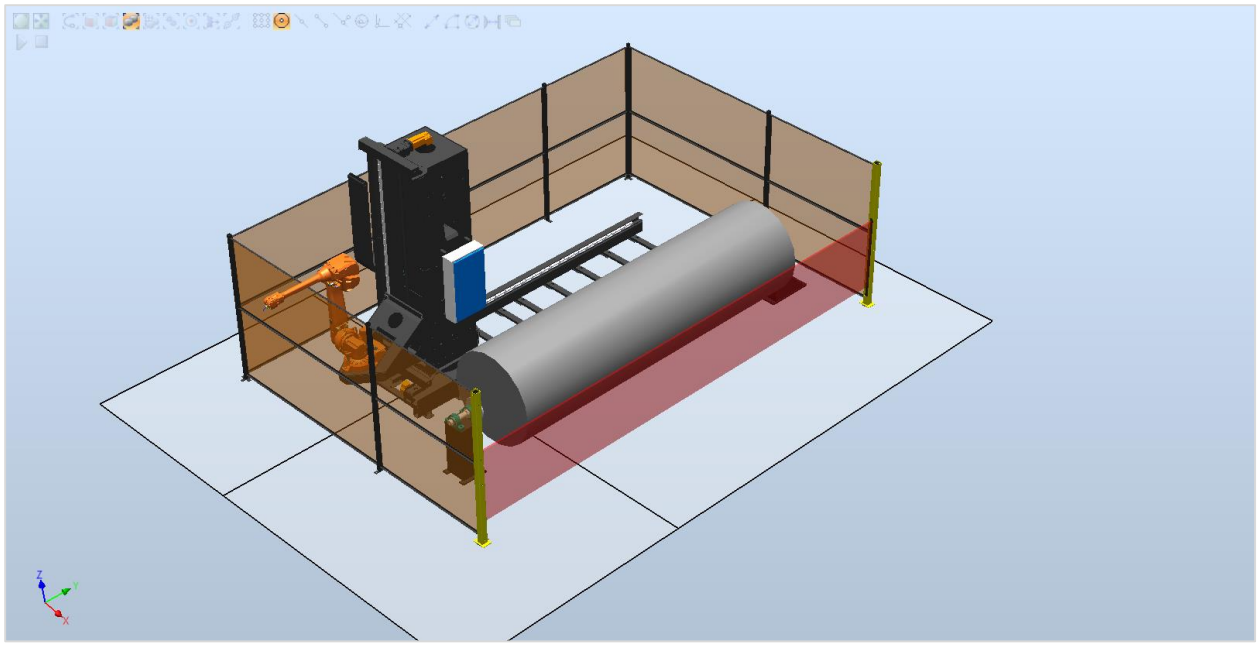


Figure 62: Automatic Station Builder Example 1

Another example of a generated station can be seen in Figure 63. This station was created for a 30,000Kg part with a diameter of 2,150mm and length of 14,000mm. This cell differs from the previous in a few major ways. First a longer track has been loaded to accommodate the length of the new part. This mechanism has also been changed so the gantry can move the total length of the track. New Head Stock and Tail Stock capacities were loaded to accommodate the weight. The Head Stock and Tail Stock were also placed on higher risers to allow for free rotation of the part. This additional height provided a need for a larger gantry so the robot could perform welds on the top of the part. The higher gantry also created a need for higher light curtains and safety fences, which have also been loaded.

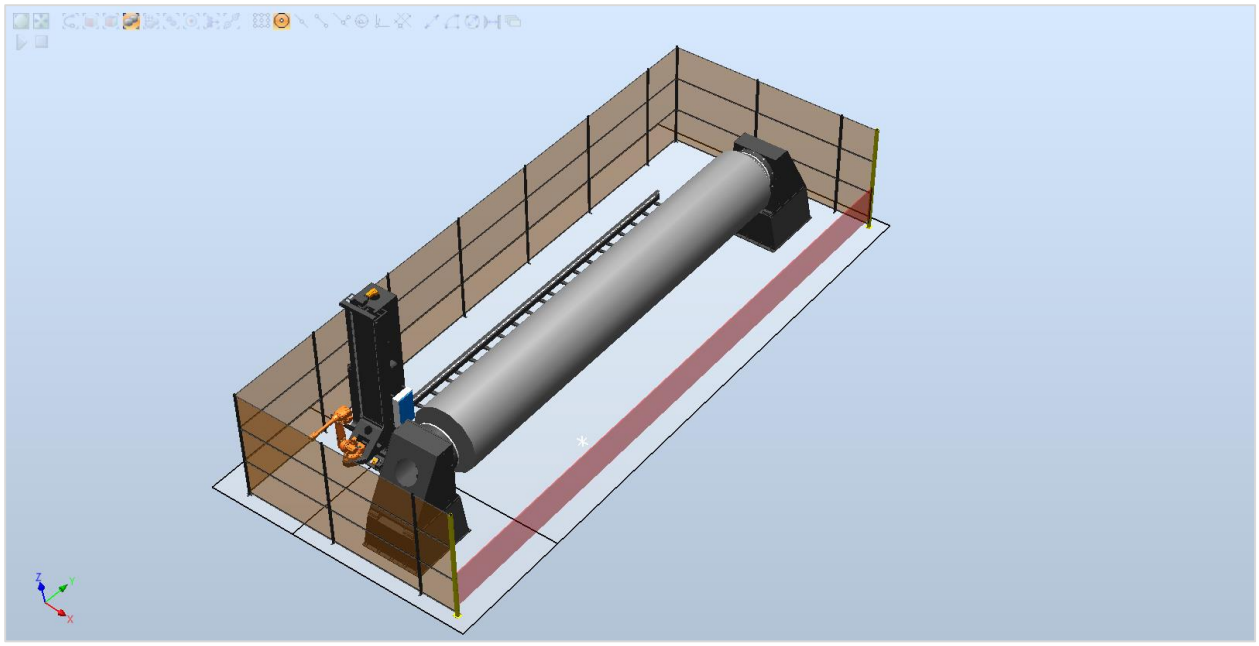


Figure 63: Automatic Station Builder Example 2

The final demonstration station can be seen in Figure 64. This station was created for a 60,000Kg part with a diameter of 3450mm, and part length of 9000mm. This part has a much larger diameter than the previous parts. The robotic welding cell accommodates for the larger diameter part by placing the Head Stock and Tail Stock further from the gantry. This also caused the station to load in two longer fence segments and place the light curtain farther.

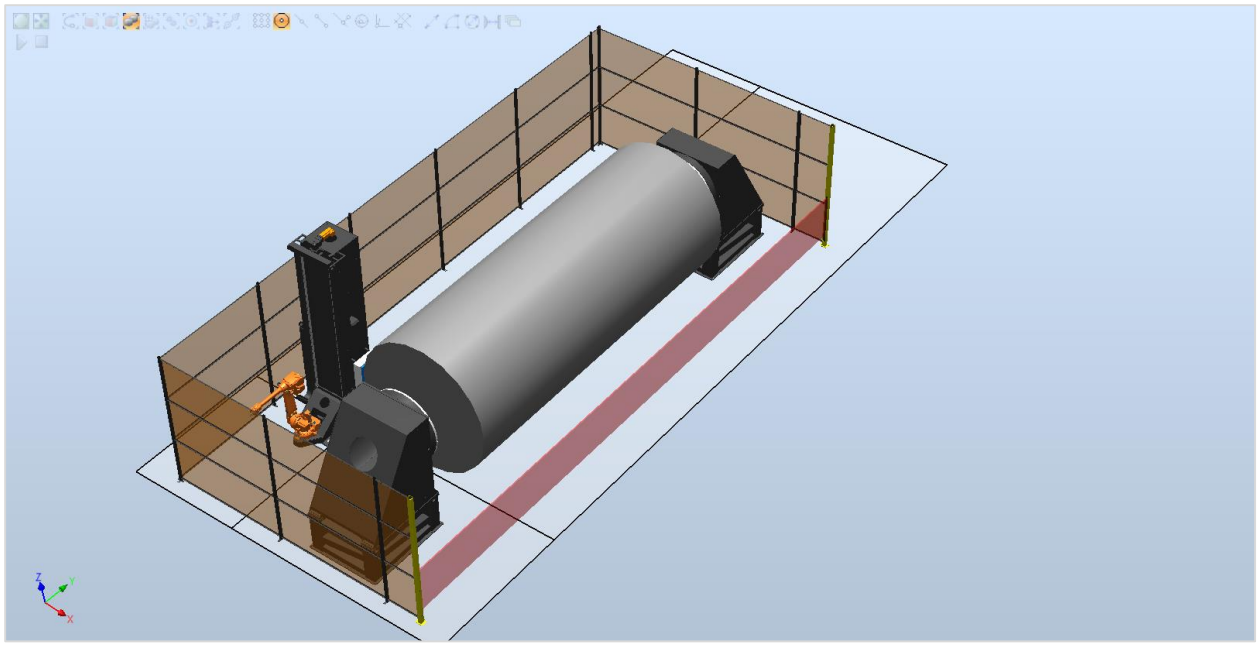


Figure 64: Automatic Station Builder Example 3

V – CONCLUSION

V.1 – Statement of Results

To increase the speed of offline programming, a series of smart components and an add-in was created. Each smart component contained the necessary geometry to create the graphical component. The smart components have the ability to create and modify mechanisms so the graphical components will be visually representative of the function performed by the mechanism. The smart components also have the capability of loading non-essential geometry and attaching it to the proper frame in order to have the station as accurate as possible during reach studies and offline programming.

The smart component series is made of three parts. The first is the automatic fence builder, which allows a user to select the desired length and height of the fence and builds it in the station. The second smart component is the track builder. This smart component loads the desired length of track; gantry, based on selected height; and boom length. The user also has the option to load in accessories and have them automatically placed and attached to the proper position on the gantry. This smart component also generates the gantry mechanism based on the height of the tower selected and the travel track length. The final smart component is the automatic light curtain component. This component allows a user to create multiple light curtains and automatically build a virtual laser fence that sits between the two light curtain posts.

An additional component was created to completely automate the construction of a robotic welding cell. The Automatic Station Builder constructs a welding cell given the major attributes of a part: weight in Kg, Diameter in mm, and Length in mm. This component leverages the smart component series to generate the appropriate robot and part positioner and includes safety equipment such as the fencing and light curtains. All components are autonomously sized and placed to construct a welding cell capable of welding the specified part.

An add-in was created in Microsoft Visual Studio C# to allow a user to quickly load a positioner and the desired riser. The positioner loads into the correct position for calibration of a virtual controller. The Head Stock and Tail Stock are selected based on desired capacity and riser height. SkyHooks are selected based on capacity, drop, and throw. The appropriate riser is loaded based on the larger of the drop and throw measurements. Additionally, Drop Centers are loaded given capacity, drop, throw, and riser height.

The automatic smart components series, additional automation component, and positioner add-in works as intended. The libraries are extremely useful and time efficient when building and updating a concept cell in ABB RobotStudio. Originally, the Wolf Robotics parts and positioners were spread between 190 individual folders, and took up almost 7 Gigabytes of computer memory. Through the use of the smart components and the add-in, 2,428 unique mechanisms can now be accessed from 8 simple menus—excluding accessory options.

Generating the mechanisms in this way not only saves the user time in the development of a concept station, but also saves substantial computer memory. The method of pre compiling libraries, used before, duplicated geometry in each mechanism. The new method uses a single piece of geometry that can be shared between multiple mechanisms. The memory used by the libraries was decreased by 80%.

The RobotStudio SDK is extremely powerful in the creation and modification of stations. Many functions that users can leverage to create concept cells are also available in the SDK. This allows knowledge from standard cell creation to be brought into the SDK to create programs to complete tasks that are preformed often.

The user manuals and maintenance manuals for the smart components and add-in have been included in the Appendices found below. These manuals cover the initial installation of the smart components as

well as usage. The maintenance manual details how to access the back end of the smart components that were visually programmed in RobotStudio.

V.2 – Future Work

Currently the ABB PC SDK does not allow the construction of virtual controllers through the SDK. As ABB increases the functionality of RobotStudio and the SDK, a method for automatically creating controllers based on the existing mechanisms in a station would be possible. If this addition were to be made to RobotStudio, a virtual controller could be compiled and would be representative of the mechanisms that are used in the station. This would then allow a user to do full simulations immediately instead of separately after making a controller based on the content in the station they have created.

Another area of future work is in the continued development of the smart components and add-in. Wolf Robotics is a light weight and agile company. As the direction of the company changes, it is important that the smart components continue to be updated so they can continue to be used. This includes additional error handling as the software faces new situations, updates to the smart components as ABB updates RobotStudio, and the addition of new positioners as Wolf Robotics continues the development of their welding processes.

Another area of advancement could be in the complete automation of robotic concept cell creation and weld path planning. The smart component series and automation component lay the foundation for future automation work. As noted in Xiaolong Feng's paper of optimal robot placement (2), productivity was increased by 37%. This could be extended to not only include the robot placement, but also part placement. With the combination of part placement and robot placement, the possibilities are limitless.

VI – REFRENCES

1. **Bengtsson, Daniel, and Carl-Johan Rutgersson.** *Development of BoxSweeper and BoxSweeper PLC.* Göteborg, Sweden : Chalmers University of Technology, 2008.

2. *Optimal Robot Placement Using Response Surface Method.* **Xiaolong Feng, et al.** 2009, International Journal Of Advanced Manufacturing Technology 44.1/2 (2009), pp. 201-210.

3. MAN: Modern Applications News. *Using CMM Arm Slashes Robot Programming 90%.* 2004, Vol. 38, 10.

4. Mechanical Engineering. *Loading Features on Palletizing Robots.* 2011, Vol. 133, 9.

5. **ABB.** *Operating Manual: Robot Studio 5.14. Document ID: 3HAC032104-001 Revision: D.* 2008.

6. —. *Release Notes: RobotStudio SDK 5.14.* 2011.

7. **Robotics, Fanuc.** *Accompanying Training Manual: Roboguide V6.40 Rev.B.*

8. **MotoMan.** *Instruction Manual MotoVisual Componenet Library. Reg No: ME00040EN-00.* 2008.

9. **Spaak, Anders.** Parametric Fence Ver. 3. *ABB RobotApps.* [Online] May 29, 2012. http://www.abb.com/product/ap/seitp327/5dd4fbc7afcb82d2c12579ad0054401d.aspx.

10. **Thomas, Dean.** Parametric Robot Stand. *ABB RobotApps.* [Online] January 01, 2011. https://robotapps.robotstudio.com/Details.aspx?fileId=367.

11. **Ramos, Richard.** Schunk Gripper. *ABB RobotApps.* [Online] November 24, 2011. https://robotapps.robotstudio.com/Details.aspx?fileId=365.

12. **Fogbring, Simon.** Coordinate File Import 5.14 ver.2. *Abb RobotApps.* [Online] June 17, 2011. http://www.abb.com/product/ap/seitp327/5dd4fbc7afcb82d2c12579ad0054401d.aspx.

13. **Admin, ABB.** RS User Library Addin. *ABB RobotApps.* [Online] November 11, 2008. http://www.abb.com/product/ap/seitp327/5dd4fbc7afcb82d2c12579ad0054401d.aspx.

14. **ABB.** *Application Manual: FlexPendant SDK RobotWare 5.14. Document ID: 3HAC036958-001 Revision: A.* 2010.

15. —. *Application Manual: PC SDK RobotWare 5.14. Document ID: 3HAC036957-001 Revision: A.* 2010.

16. —. *Application Manual: Robot Application Builder RobotWare 5.0. Document ID: 3HAC028083-001 Revision: D.* 2008.

17. **MotoMan.** *MotoVisual 3D Simulation Software Quick Start Guide 2007.* 2007.

18. ABB System Programs Robots in Parallel. March 2006, Vol. 31, 3, pp. 45-45.

APPENDIX A. USER MANUAL

Smart Components

Automatic Fencing

1. Import smart component as library.

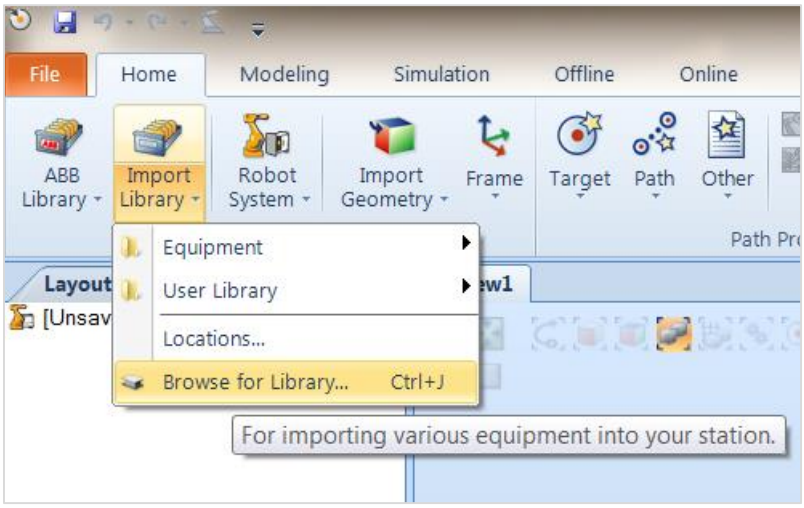


Figure 65: User Manual Import Fencing

2. A message will appear asking to verify the smart component—click *Yes.*

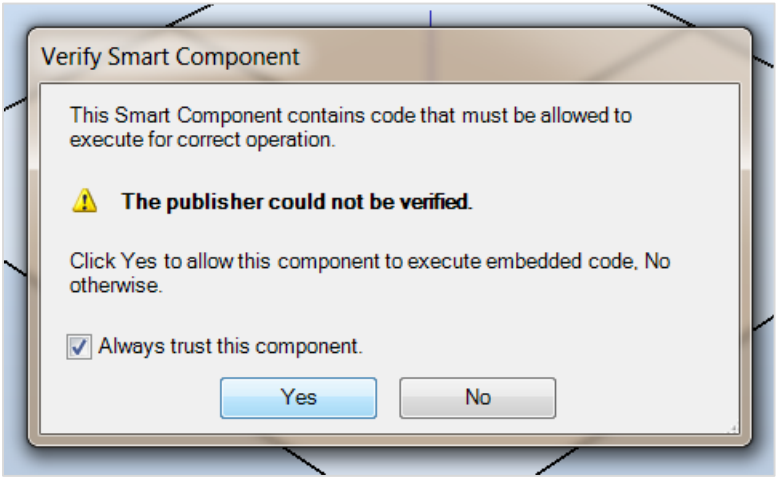


Figure 66: User Manual Verify Component

3. In the *Home* tab under layout, double-click *Automatic_Fencing* to bring up selection menu.

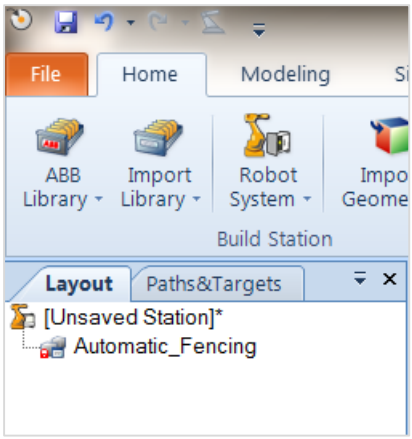


Figure 67: User Manual Smart Component Selection

4. In the selection menu, select desired height and length of fence in feet.

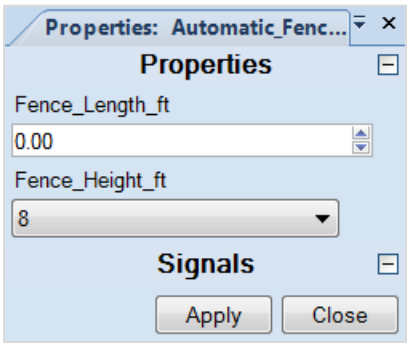


Figure 68: User Manual Edit Fence Height & Length

5. In the selection menu, press *Apply* to activate the smart component.

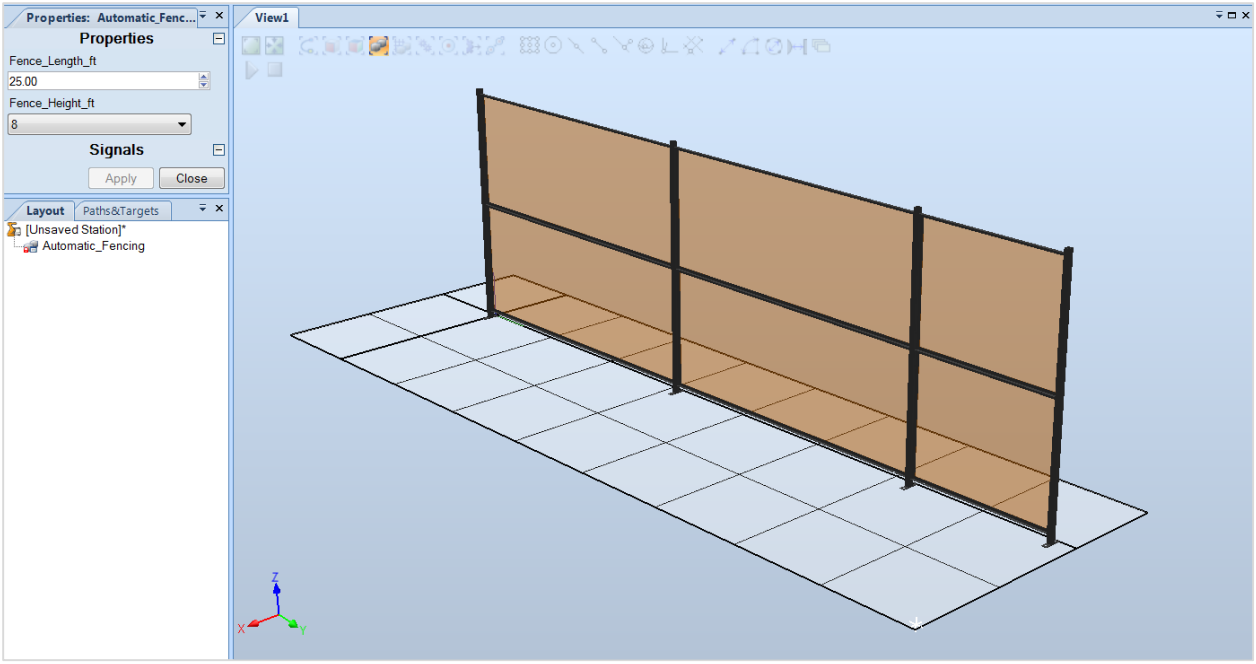


Figure 69: User Manual Fencing Final

Automatic Track Builder

1. Import smart component as library.

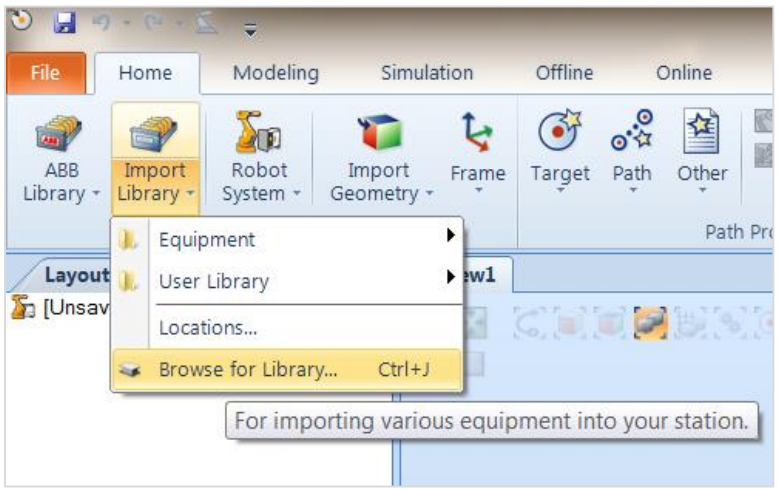


Figure 70: User Manual Import Travel Track

2. A message will appear asking to verify the smart component—click *Yes.*

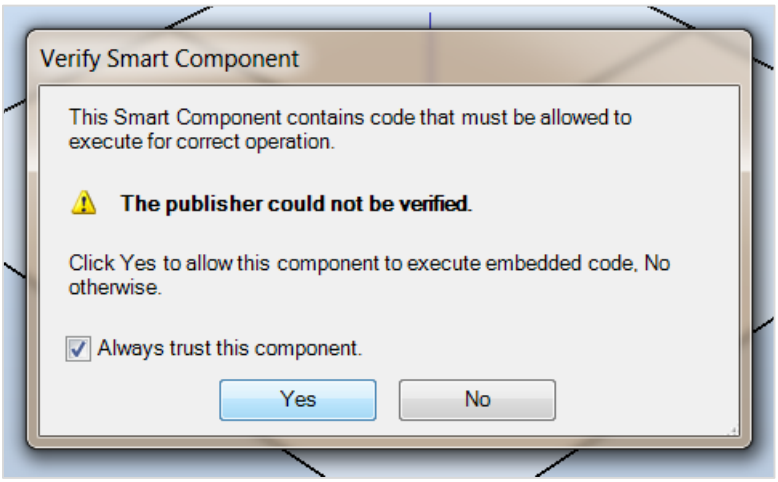


Figure 71: User Manual Verify Component

3. In the *Home* tab under layout, double-click *Automatic_Track* to bring up selection menu.

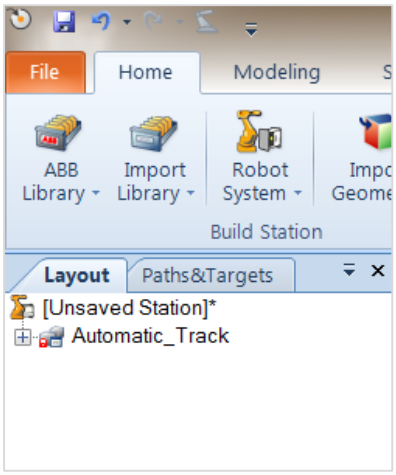


Figure 72: User Manual Smart Component Selection

4. In the selection menu, select Track Length, Gantry Type, Boom Length or Track Length and RTT (Standard Robot Carriage).

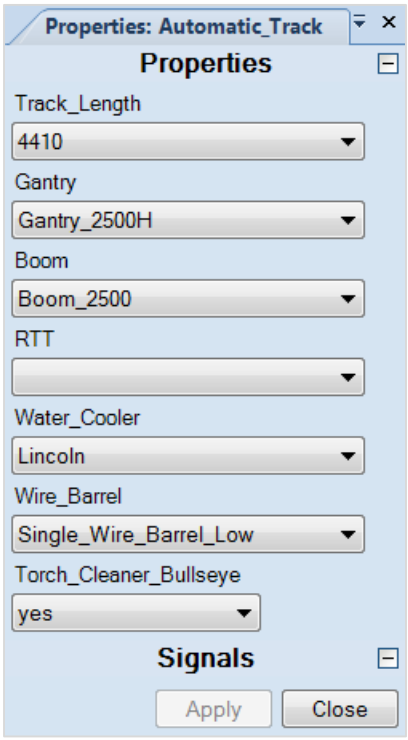


Figure 73: User Manual Edit Travel Track Options

5. In the selection menu, press *Apply* to activate the smart component.

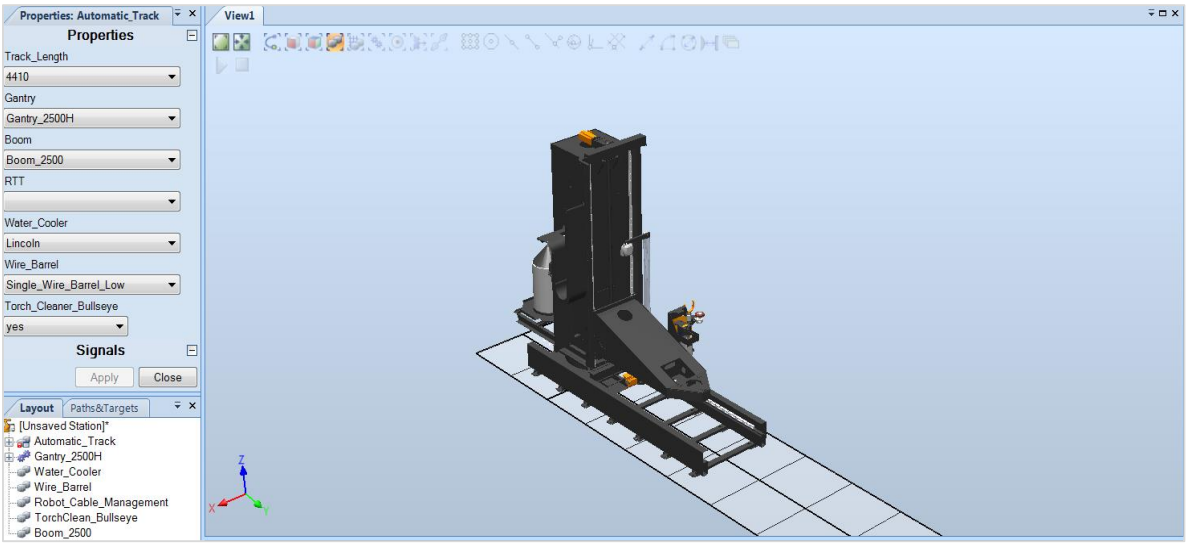


Figure 74: User Manual Final Travel Track

6. A user can now select the gantry mechanism and jog any of the three axes using mechanism joint jog.

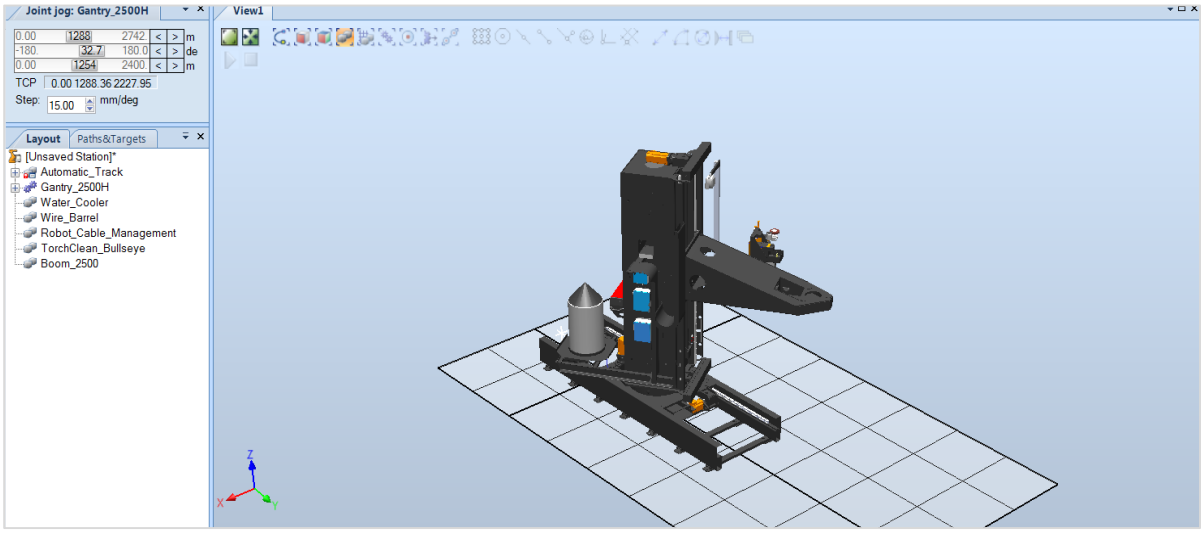


Figure 75: User Manual Jog Mechanism

Automatic Light Curtain

1. Import smart component as library.

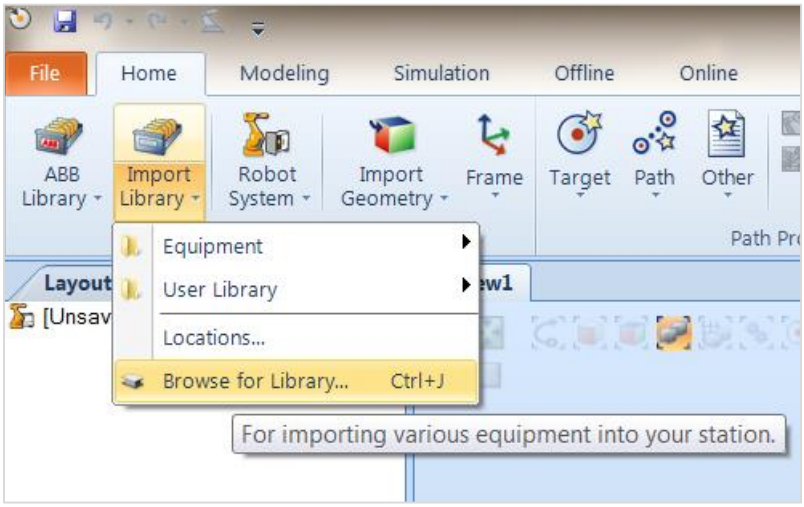


Figure 76: User Manual Import Light Curtain

2. A message will appear asking to verify the smart component—click *Yes.*

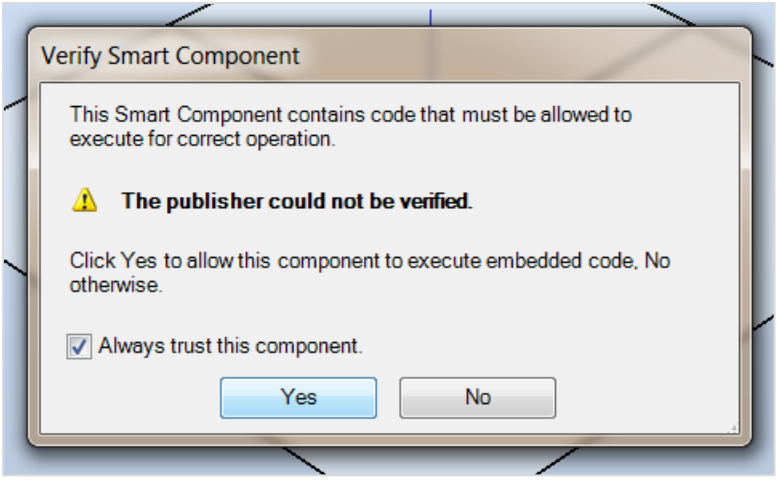


Figure 77: User Manual Verify Component

3. In the *Home* tab under layout, double-click *Automatic_Track* to bring up selection menu.

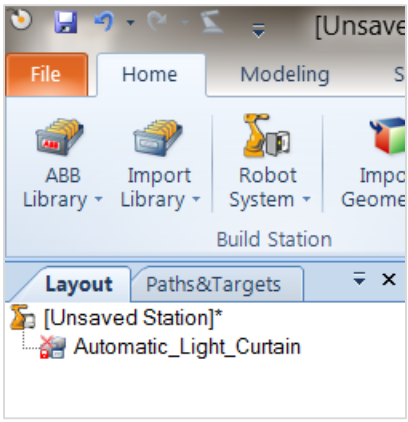


Figure 78: User Manual Smart Component Selection

4. In the selection menu, select a value for the height of the Light Curtain (up to 9).

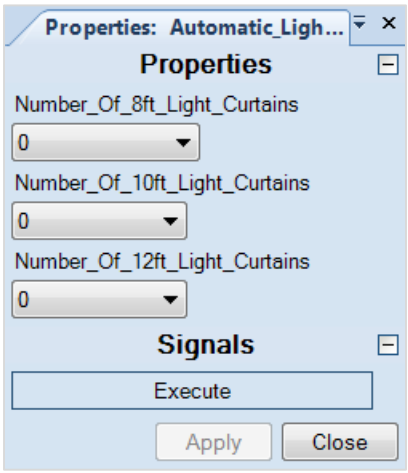


Figure 79: User Manual Edit Light Curtain Options

5. In the selection menu, press *Apply* then *Execute* to activate the smart component.
6. Now the user can select the mechanism and using mechanism joint jog increase the distance between the light poles.
7. To update the light curtain length, position, or orientation press the *Execute* button in the selection menu.

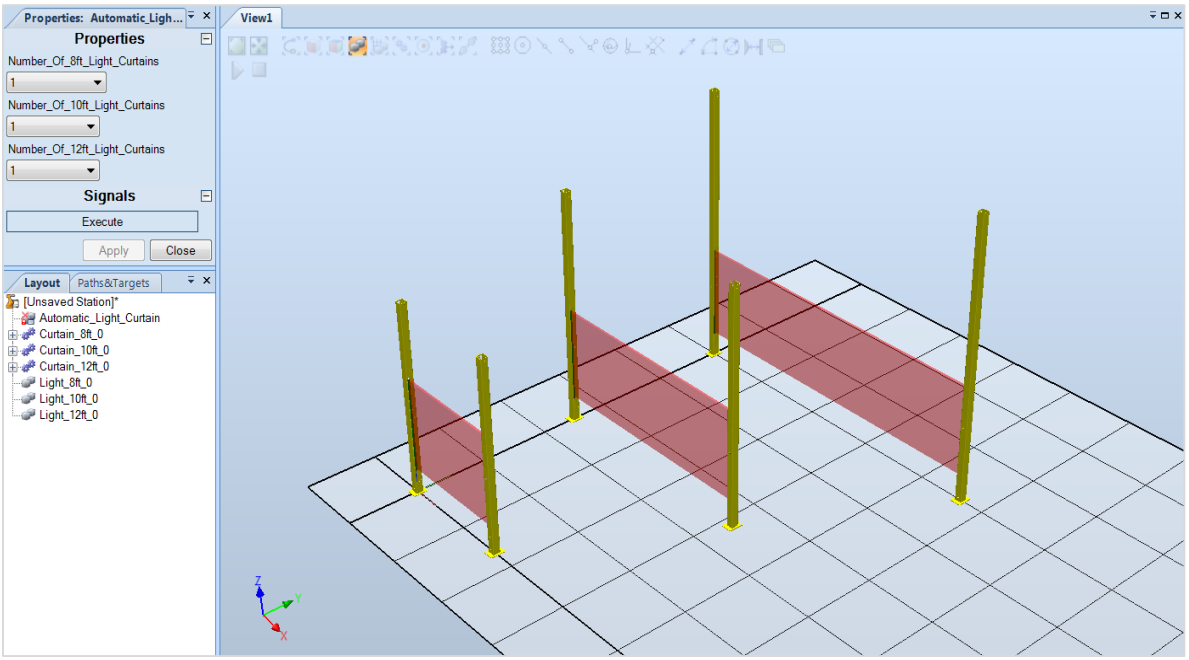


Figure 80: User Manual Light Curtain Final

Positioners Add-in

Installing Positioners Add-in

1. Begin by executing the installer for the Wolf Positioners.

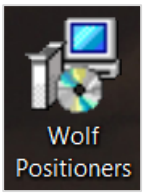


Figure 81: Install Icon

2. At the welcome window for the Wolf Positioners Setup—click *Next.*

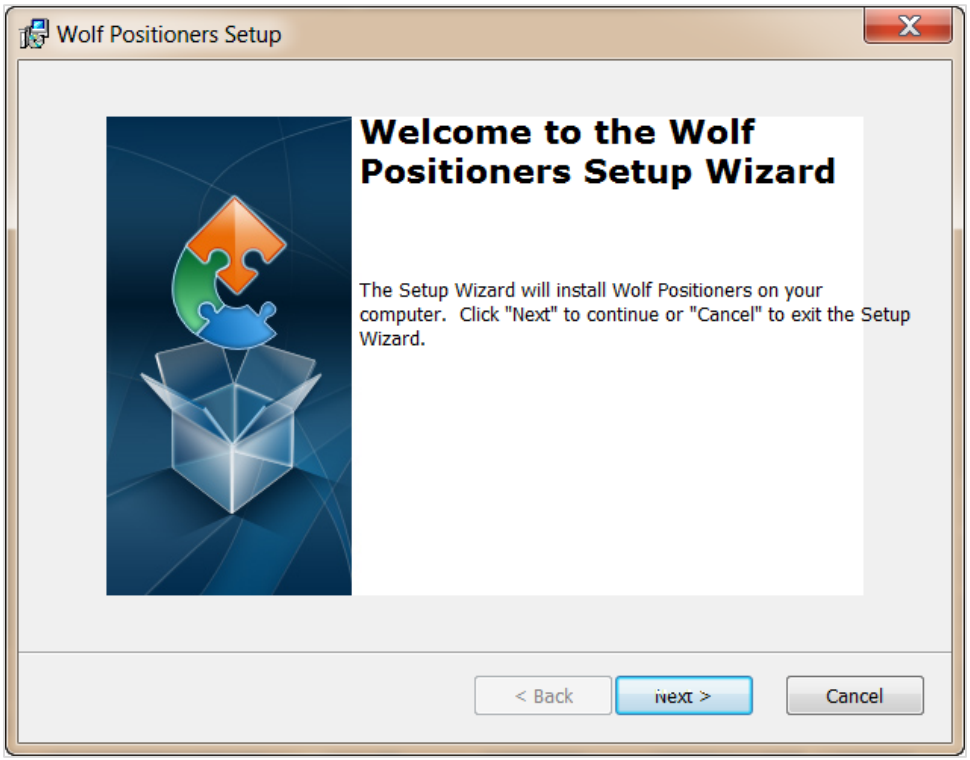


Figure 82: Install Welcome Screen

3. Leave the default installation folder in the next window and click *Next.*

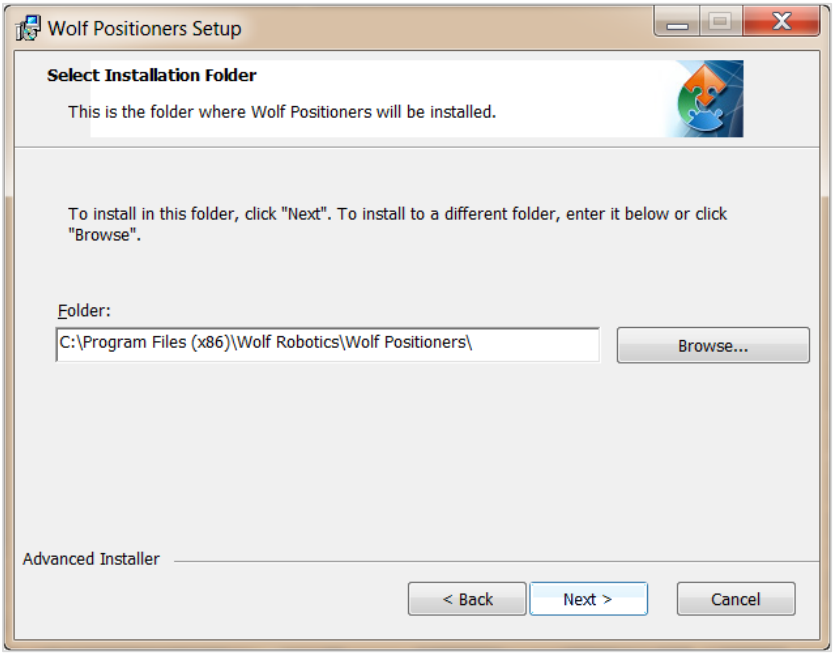


Figure 83: Install File Path

4. In the Ready to Install window, click *Install.*

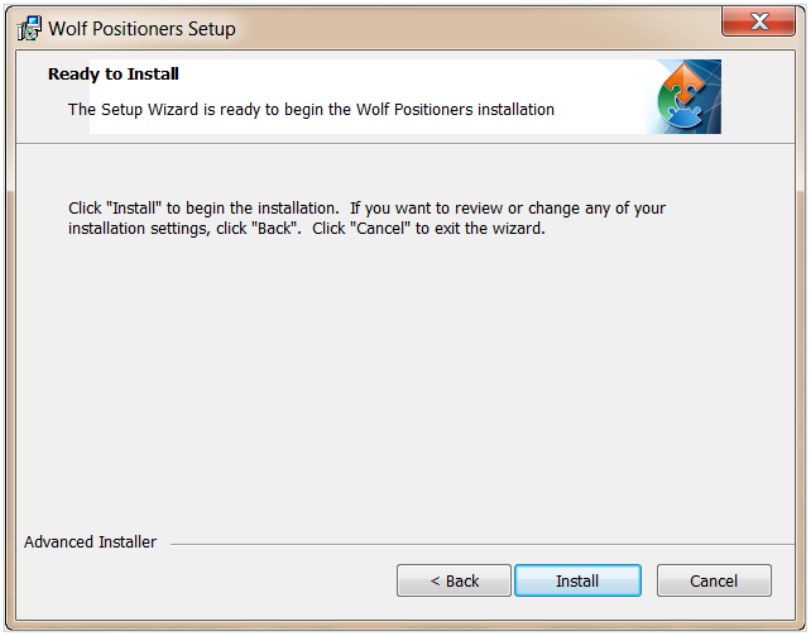


Figure 84: Begin Install

SkyHook

1. After installing the Positioners add-in, a new tab called Wolf Positioners will be available. Go to this tab and click the *Create SkyHook* button.

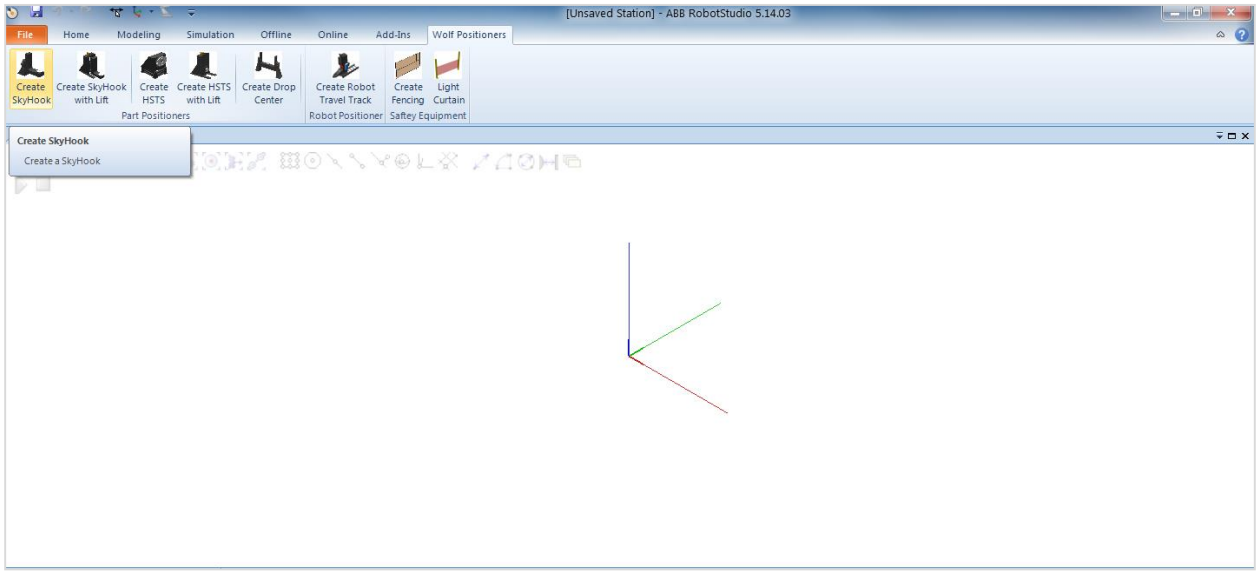


Figure 85: User Manual Create SkyHook

2. This will open up a new window for selecting SkyHook options. *Throw* and *drop* will automatically update with the standard libraries once a capacity has been selected. After the user has selected *capacity, throw,* and *drop,* click the *Load* button.

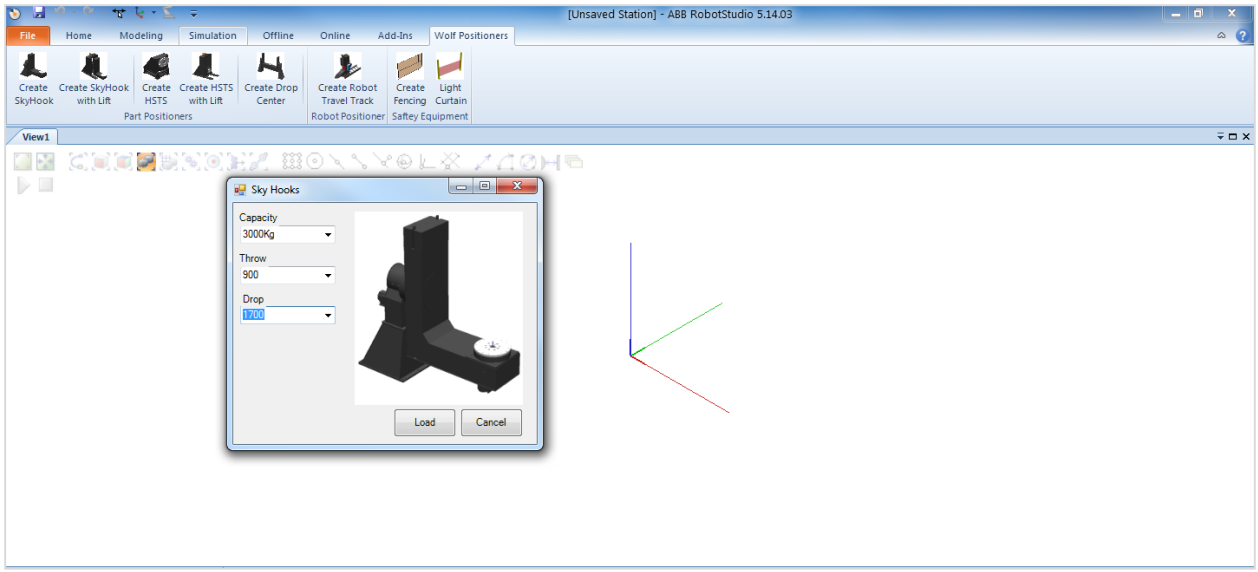


Figure 86: User Manual SkyHook Selection Window

3. This will load the desired SkyHook into the station. The SkyHook can now be connected to a controller or manually moved using the mechanism joint jog command.

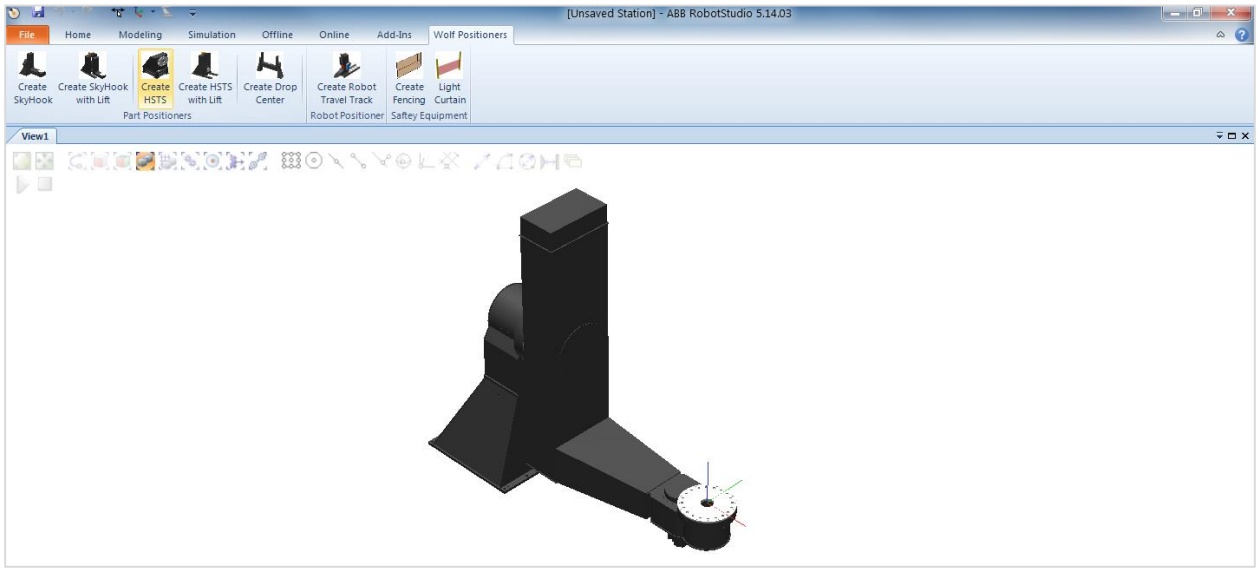


Figure 87: User Manual SkyHook Final

SkyHook with Lift

1. After installing the Positioners add-in, a new tab called Wolf Positioners will be available. Go to this tab and click the *Create SkyHook with Lift* button.

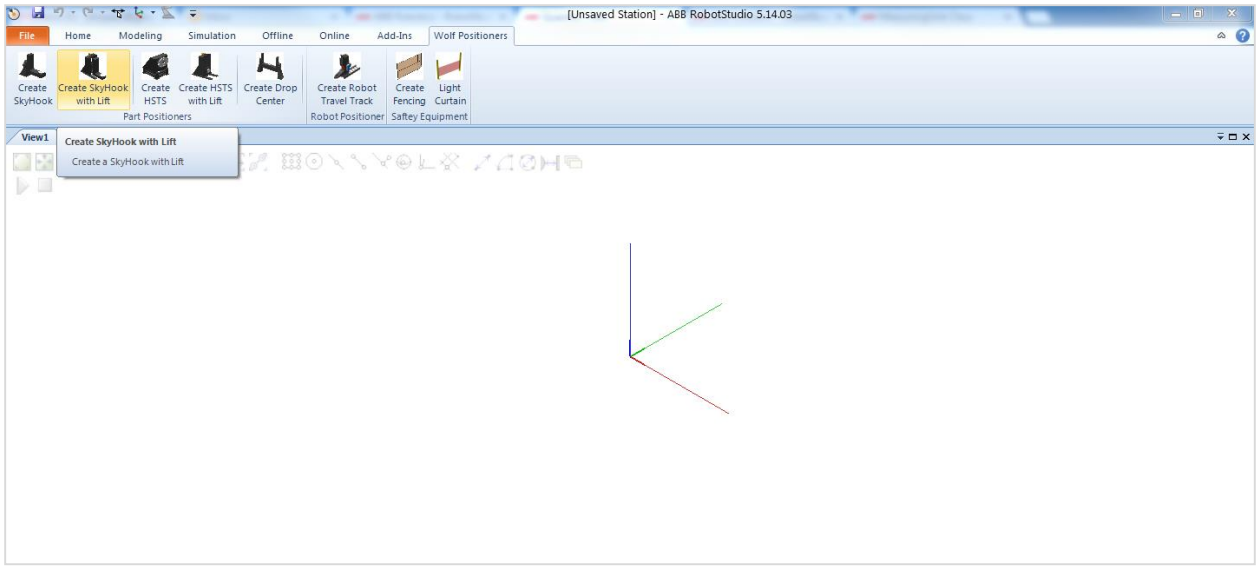


Figure 88: User Manual Create SkyHook with Lift

2. This will open up a new window for selecting SkyHook with Lift options. *Throw* and *drop* will automatically update with the standard libraries once a capacity has been selected. After the user has selected *capacity, throw,* and *drop,* click the *Load* button.

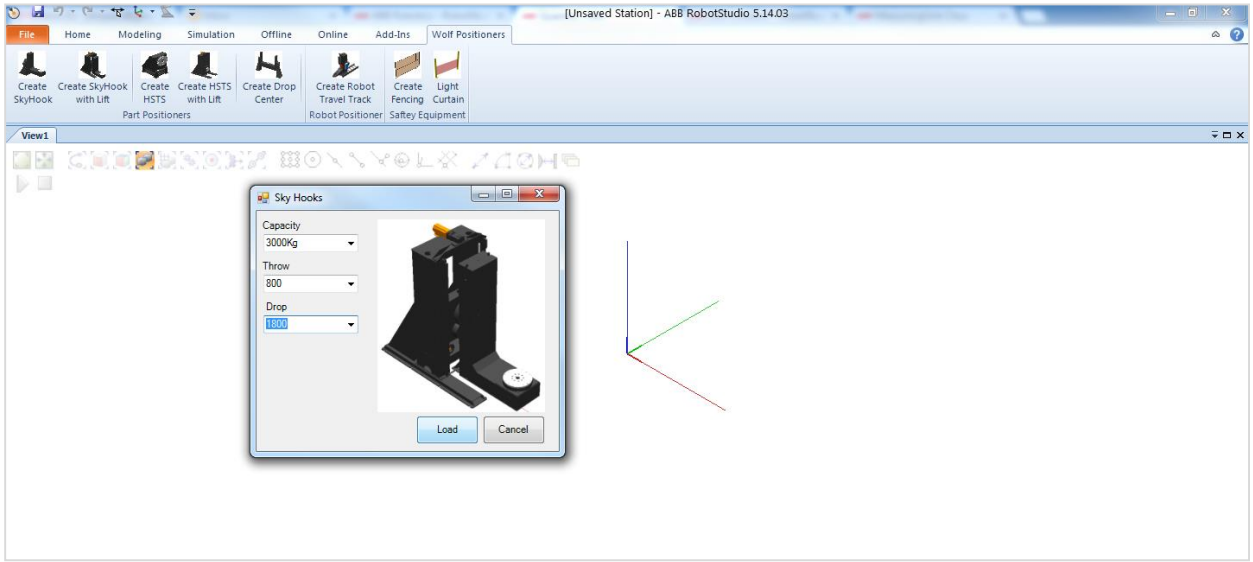


Figure 89: User Manual SkyHook with Lift Selection Window

3. This will load the desired SkyHook into the station. This SkyHook can now be connected to a controller or manually moved using the mechanism joint jog command.

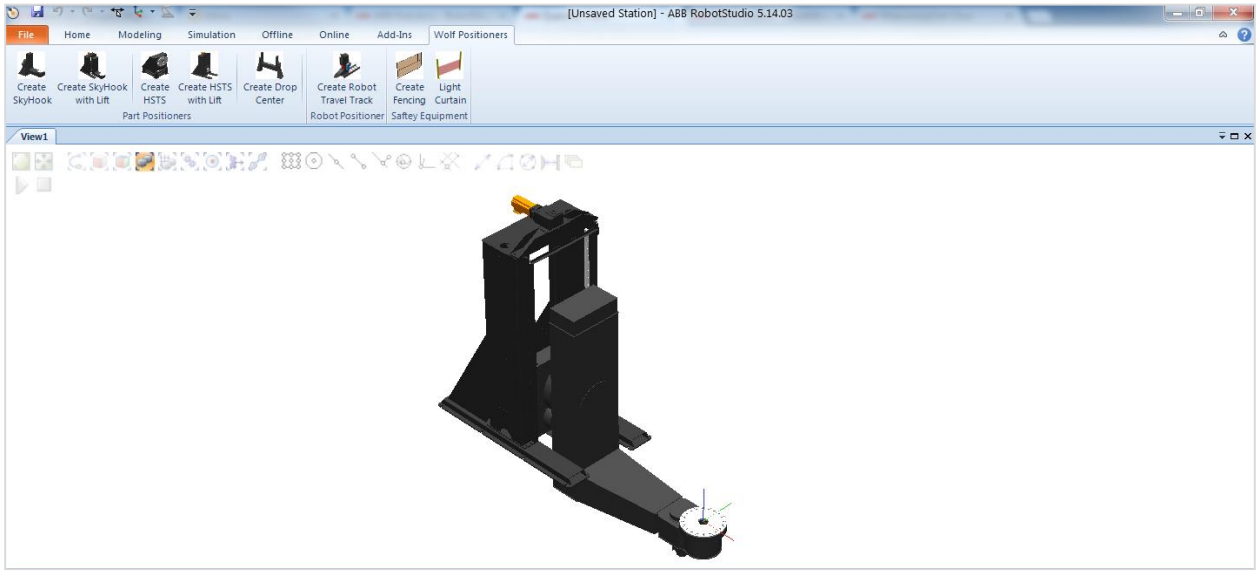


Figure 90: User Manual SkyHook with Lift Final

Head Stock Tail Stock

1. After installation of the Positioners add-in has completed, a new tab called Wolf Positioners will be available. Go to this tab and click the *Create HSTS* button.

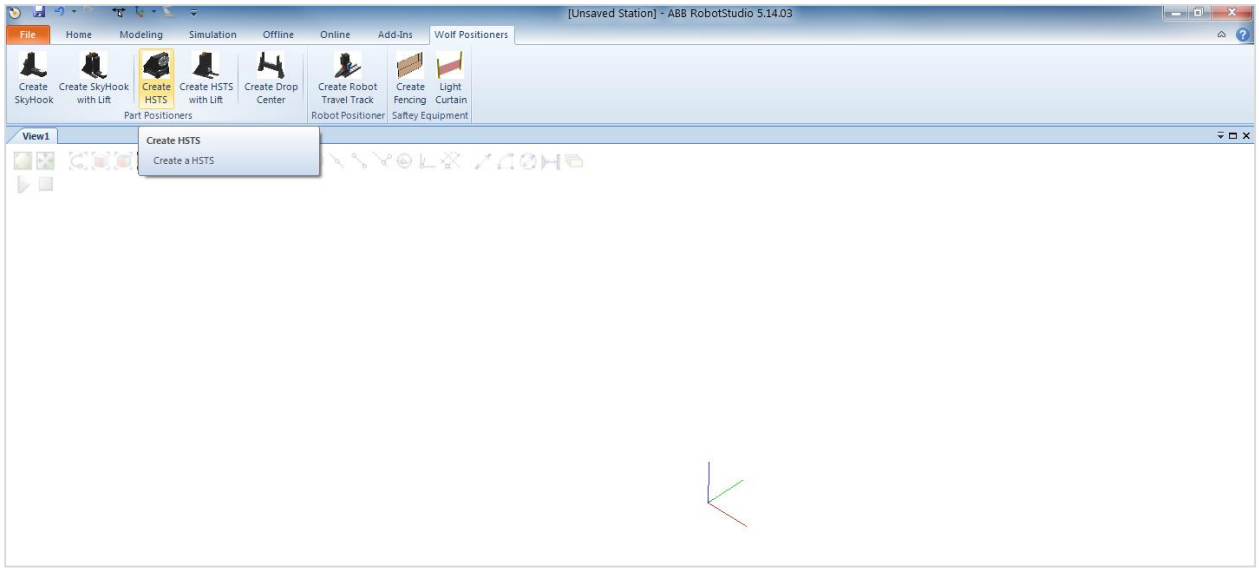


Figure 91: User Manual Create HSTS

2. This will open a new window where the Head Stock and Tail Stock options can be selected. After the user has selected capacity and riser height, click the *Load* button.

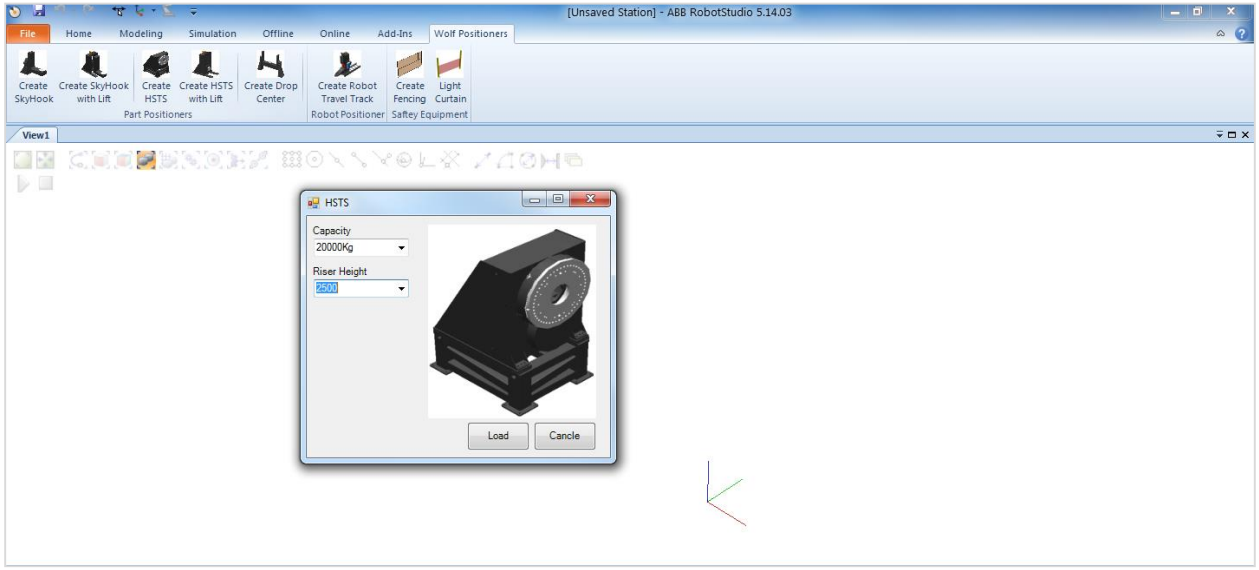


Figure 92: User Manual HSTS Selection Window

3. This will load the desired Head Stock and Tail Stock into the station. This Head Stock and Tail Stock can now be manually moved to an appropriate distance.

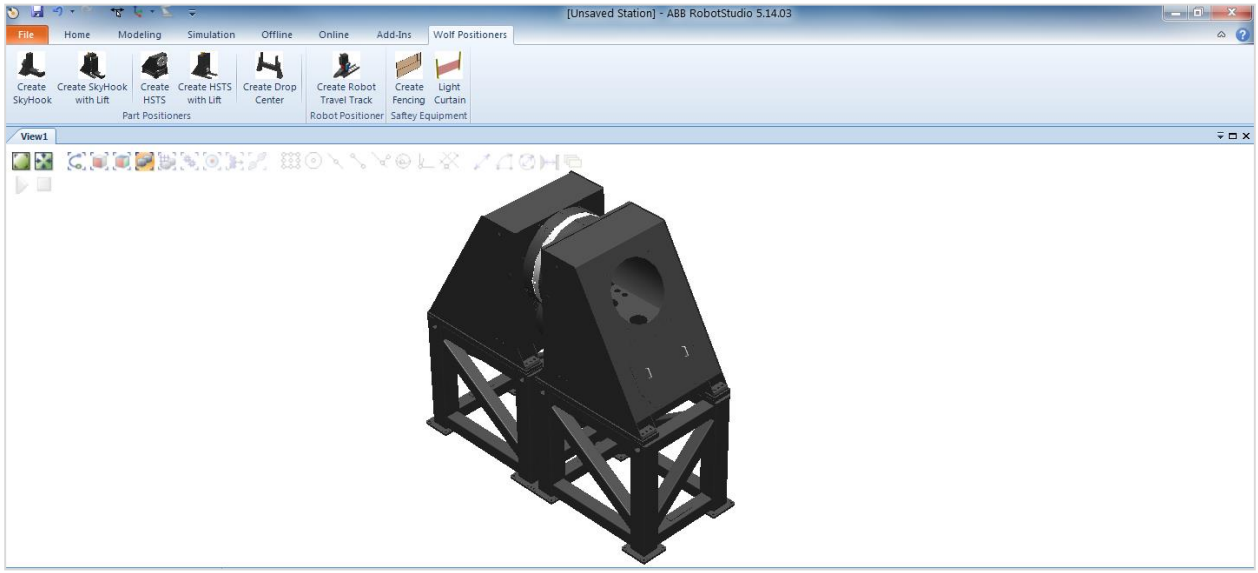


Figure 93: User Manual Loaded HSTS

4. The Head Stock can now be connected to a virtual controller.

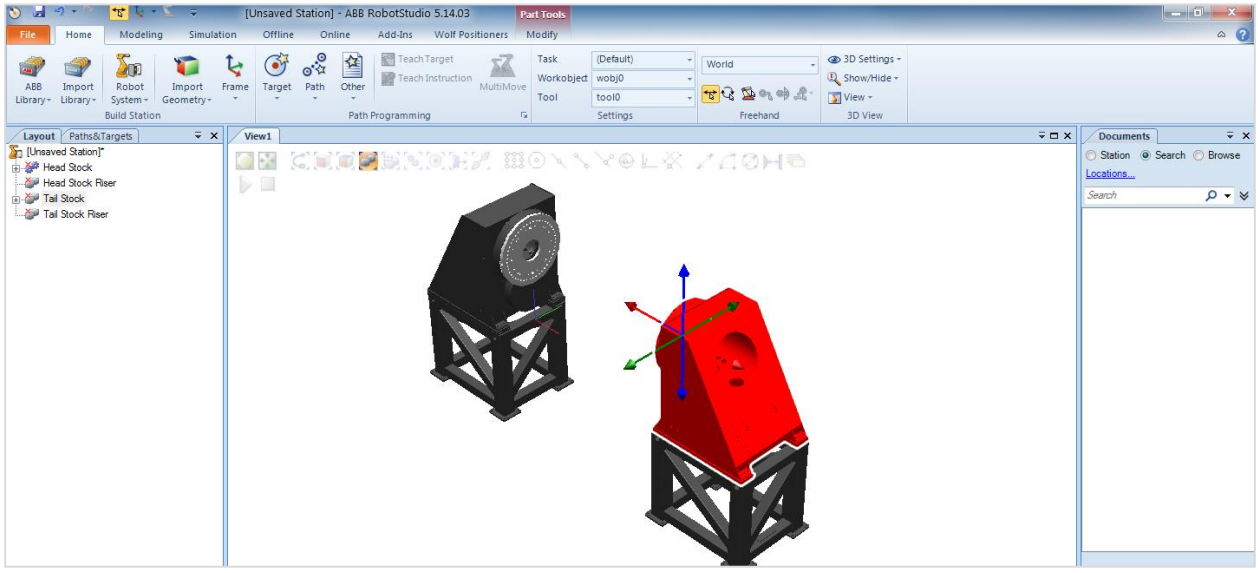


Figure 94: User Manual Position HSTS

Head Stock Tail Stock with Lift

1. After installation of the Positioners add-in has completed, a new tab called Wolf Positioners will be available. Go to this tab and click the *Create HSTS* with Lift button.

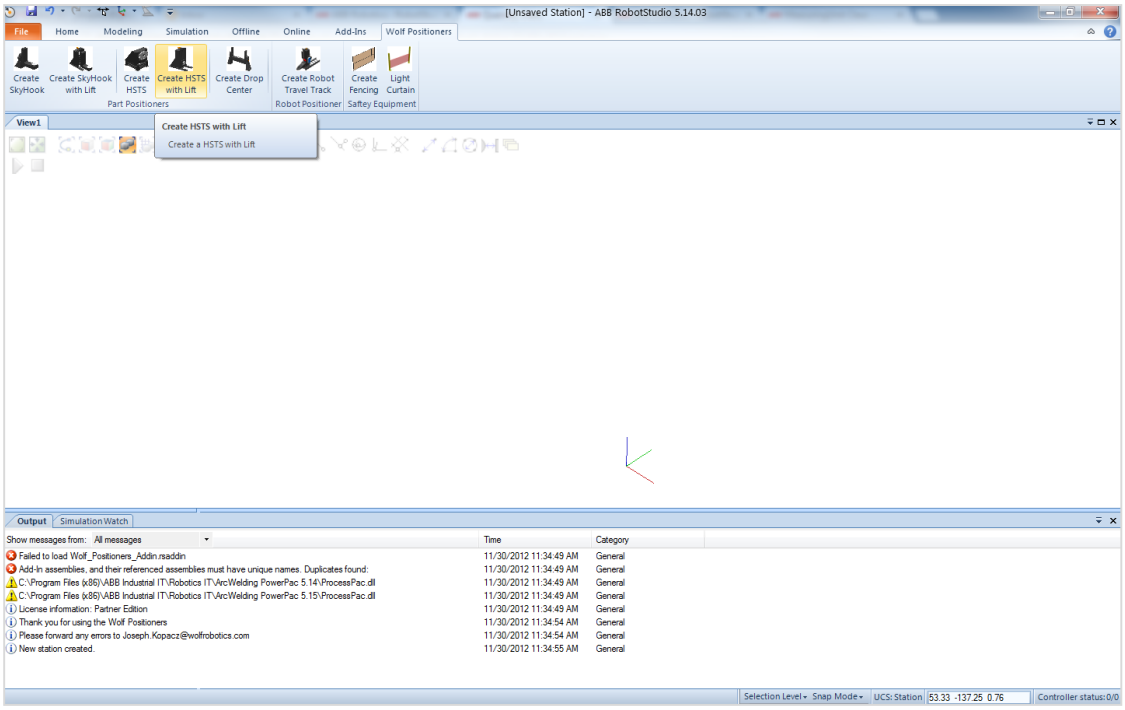


Figure 95: User Manual Create HSTS with Lift

2. This will open a new window where the Head Stock and Tail Stock options can be selected. After the user has selected capacity, click the *Load* button.

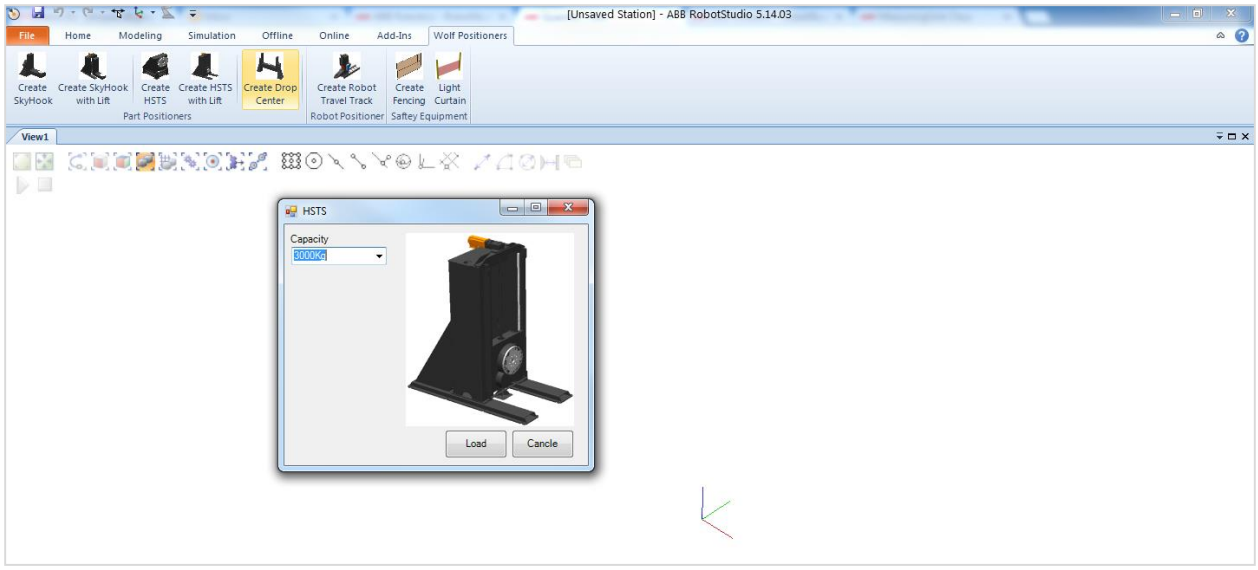


Figure 96: User Manual HSTS with Lift Selection Window

3. This will load the desired Head Stock and Tail Stock into the station. This Head Stock and Tail Stock can now be manually moved to an appropriate distance.

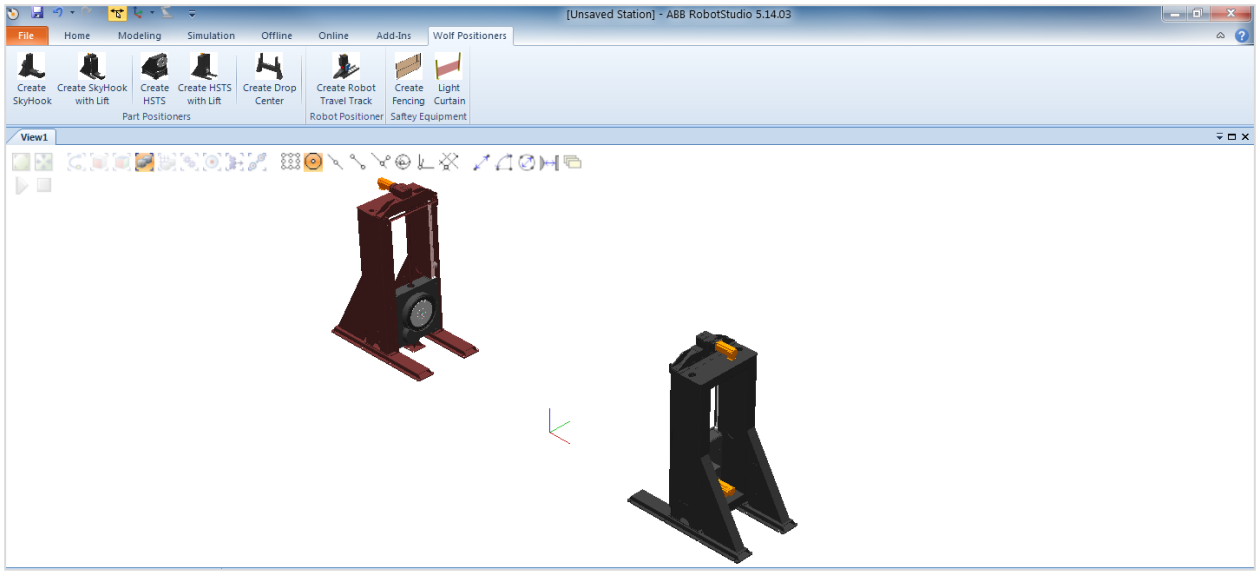


Figure 97: User Manual HSTS with Lift Final

4. The Head Stock can now be connected to a virtual controller.

Drop Center

1. After installing the Positioners add-in, a new tab called Wolf Positioners will be available. Go to this tab and click the *Create Drop Center* button.

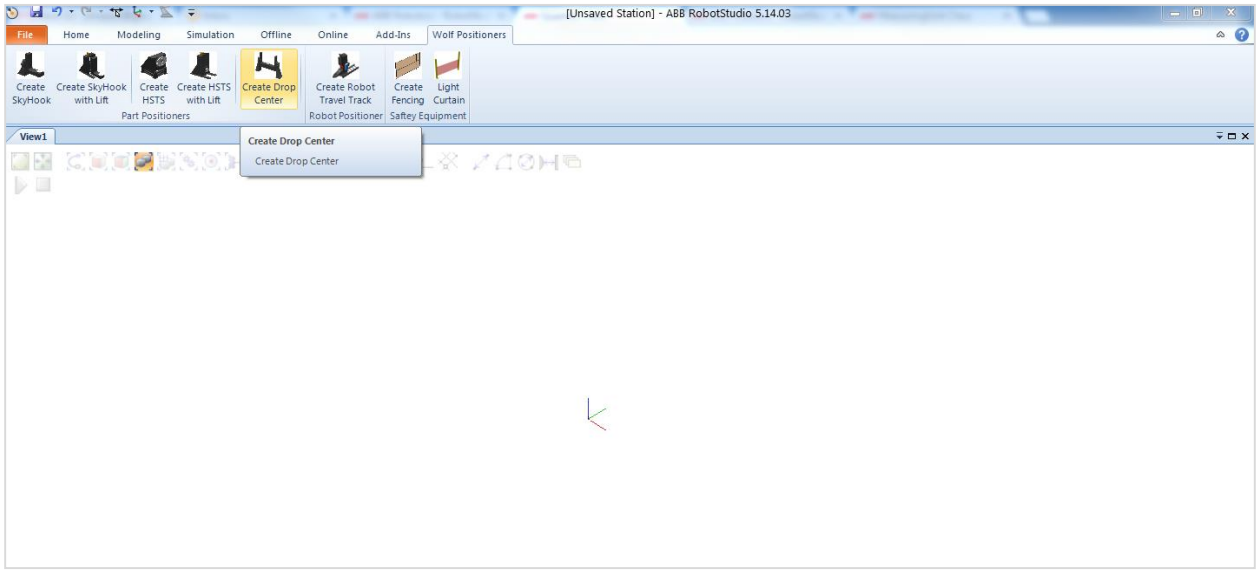


Figure 98: User Manual Create Drop Center

2. This will open a new window where the Drop Center options can be selected. After the user has selected capacity, throw, drop and riser height, click the *Load* button.

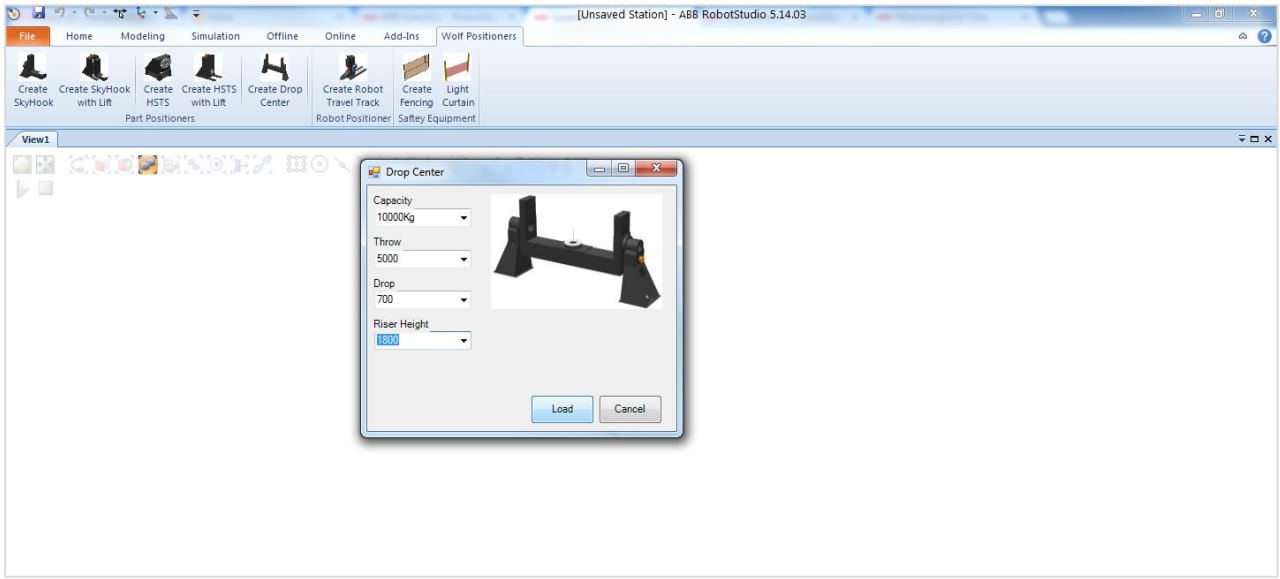


Figure 99: User Manual Drop Center Selection Window

3. This will load the selected Drop Center into the station.

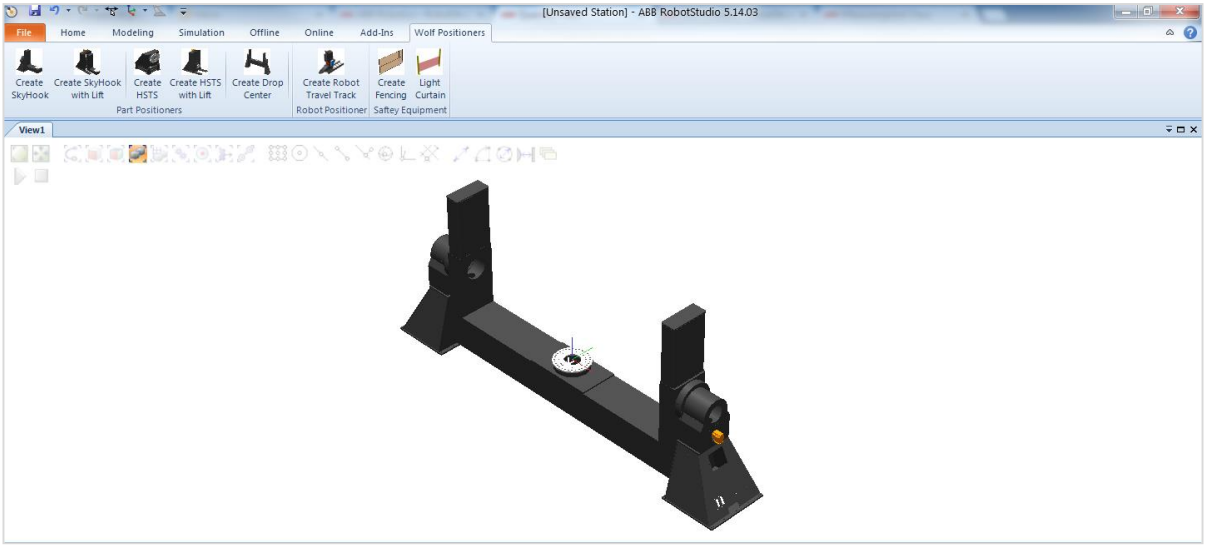


Figure 100: User Manual Drop Center Final

Automatic Station Builder

1. Import smart component as library.

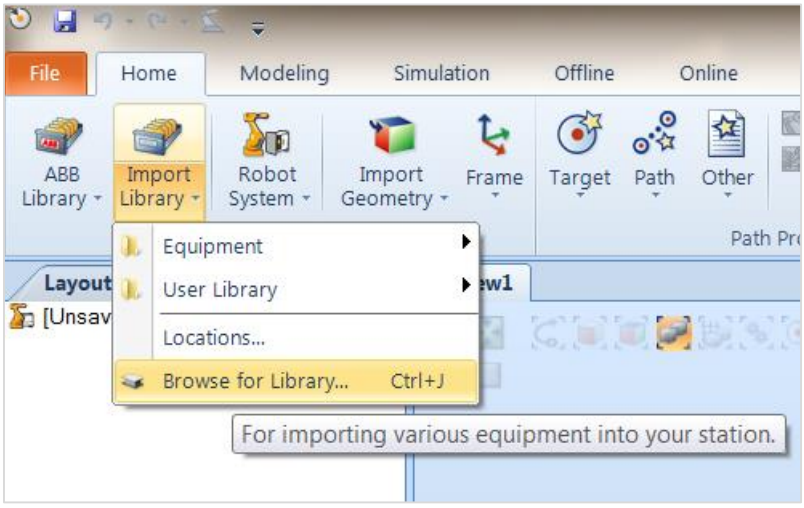


Figure 101: User Manual Import Fencing

2. A message will appear asking to verify the smart component—click *Yes.*

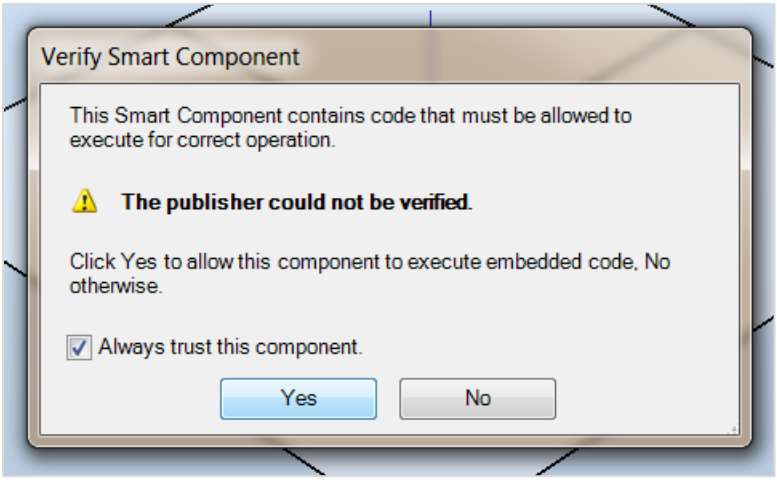


Figure 102: User Manual Verify Component

3. In the *Home* tab under layout, double-click *Automatic_Station_Builder* to bring up selection menu.

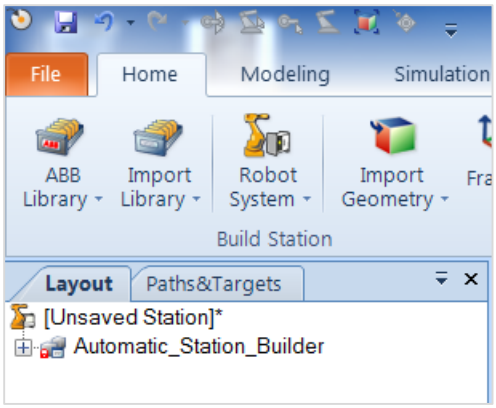


Figure 103: User Manual Smart Component Selection

4. In the selection menu, select the parts Weight in Kg, Diameter in mm, and Length in mm.

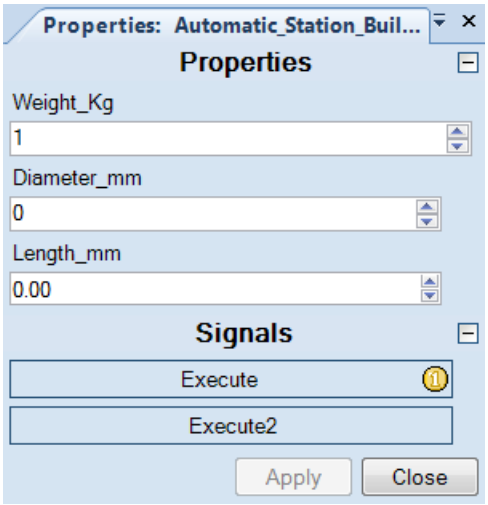


Figure 104: User Manual Edit Weight, Diameter, and Length

5. In the selection menu, press *Apply* then Execute to activate the smart component. This will bring in the components necessary to build the station.

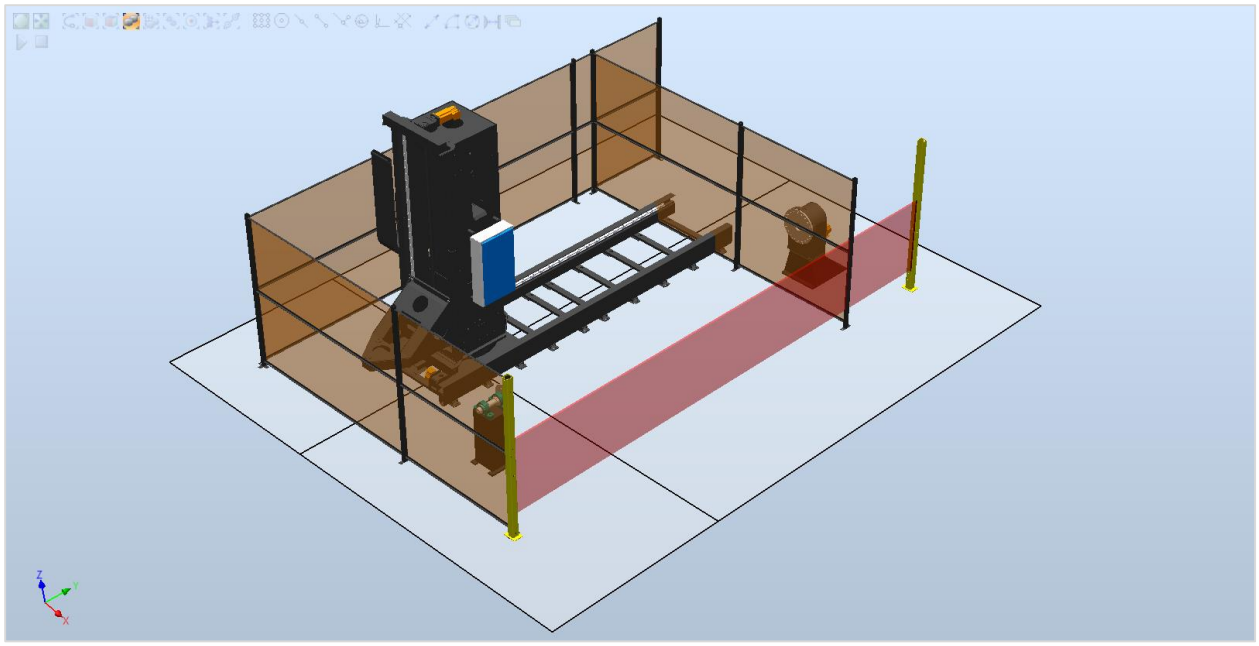


Figure 105: User Manual Execute

6. In the selection menu, press *Execute2* this will place all the loaded components.

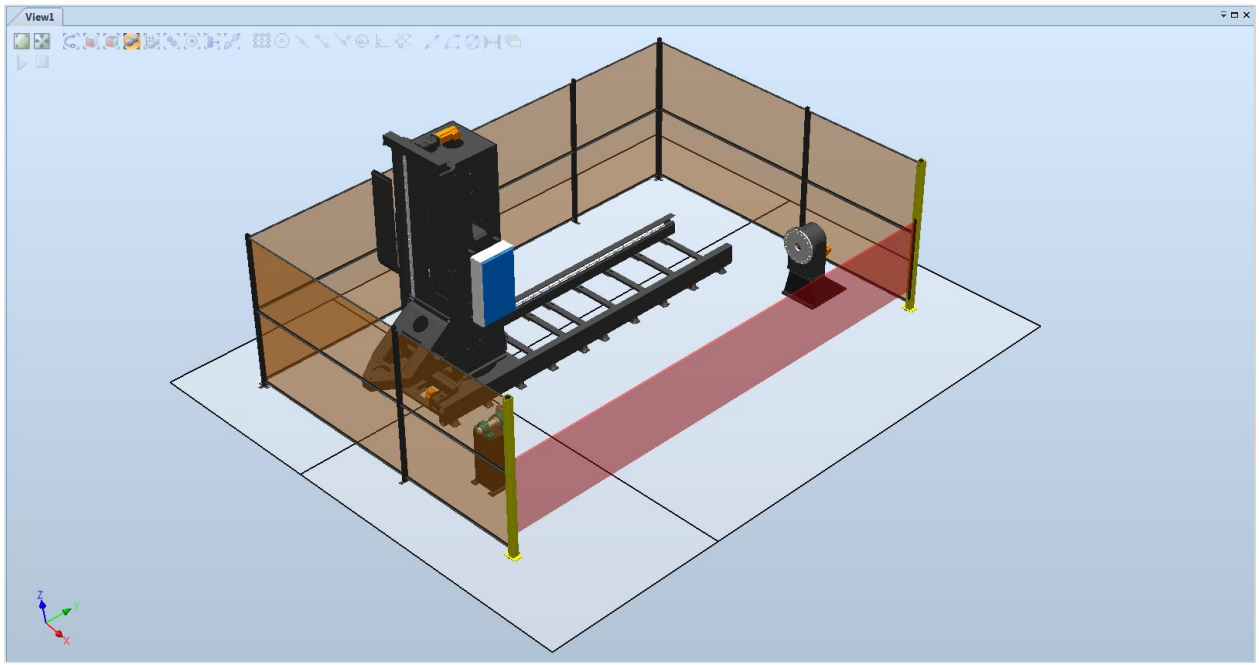


Figure 106: User Manual Execute2

APPENDIX B. SMART COMPONENT MAINTANANCE

Automatic Fencing

All of the libraries that this component uses are self-contained.

1. To access the smart component Libraries, right click on the smart component in the Layout tab and select *Disconnect Library*.

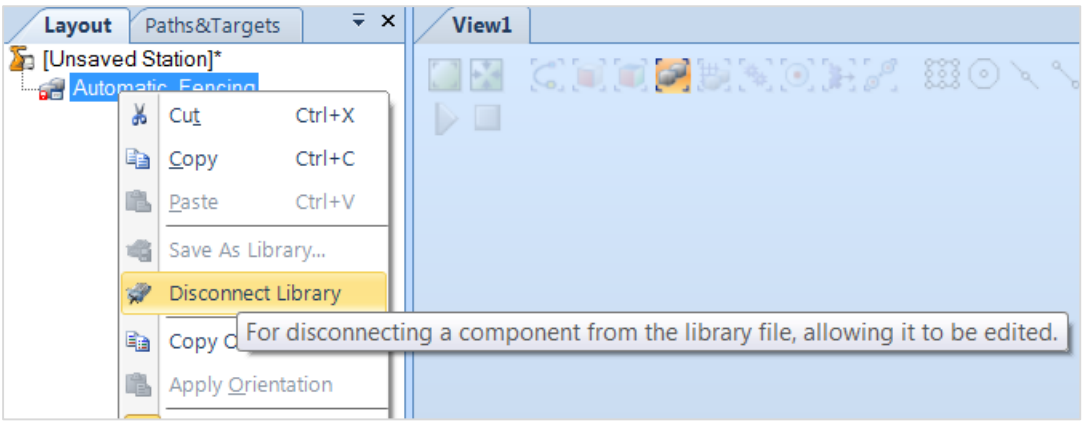


Figure 107: Maintenance Disconnect Fencing Library

2. Once the library has been disconnected, right click on the smart component and select *Edit Component*.

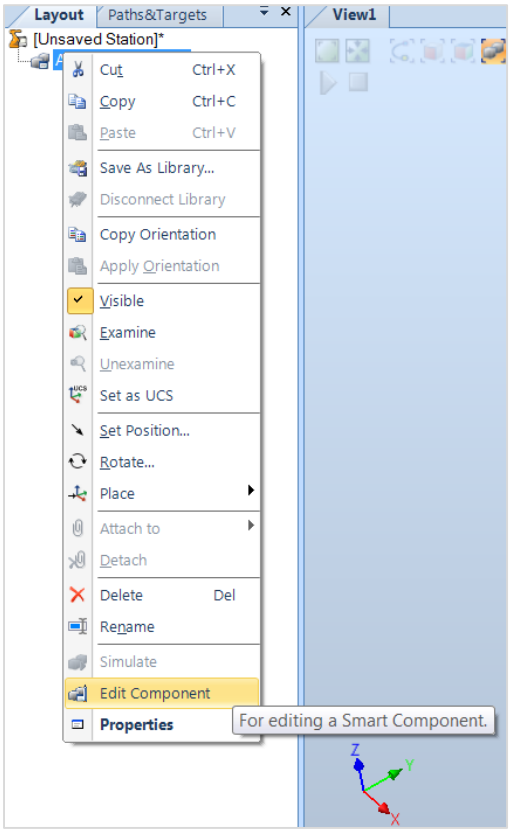


Figure 108: Maintenance Edit Fencing Component

3. This will open all of the graphic programming. For modularity this program has been separated into five sub components. Components can be selected by double-clicking on the desired component.

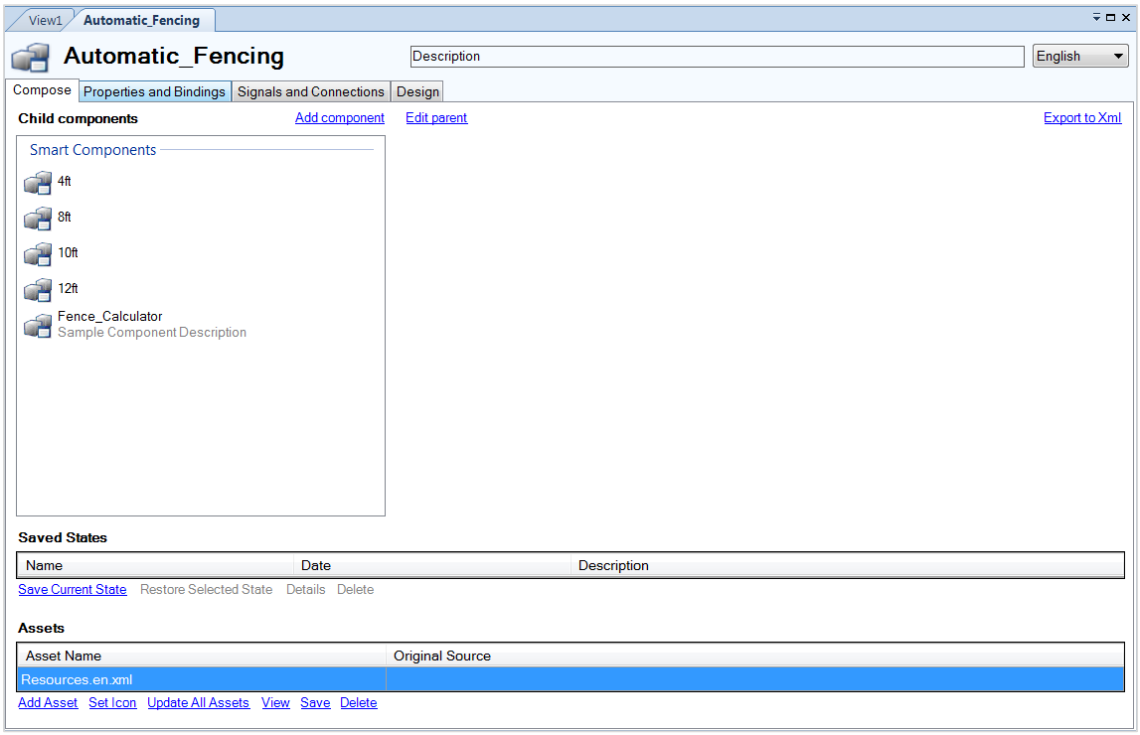


Figure 109: Maintenance Fencing Composition

4. Fence_Calculator determines the number of fence lengths needed to complete the required fencing. This function requires two inputs: Fence_Length_ft and Fence_Height_ft. The component then outputs count of each segment's height and length, distance of each height and length, and the offset necessary. More about this component can be found in the Programming of Fence_Calculator.

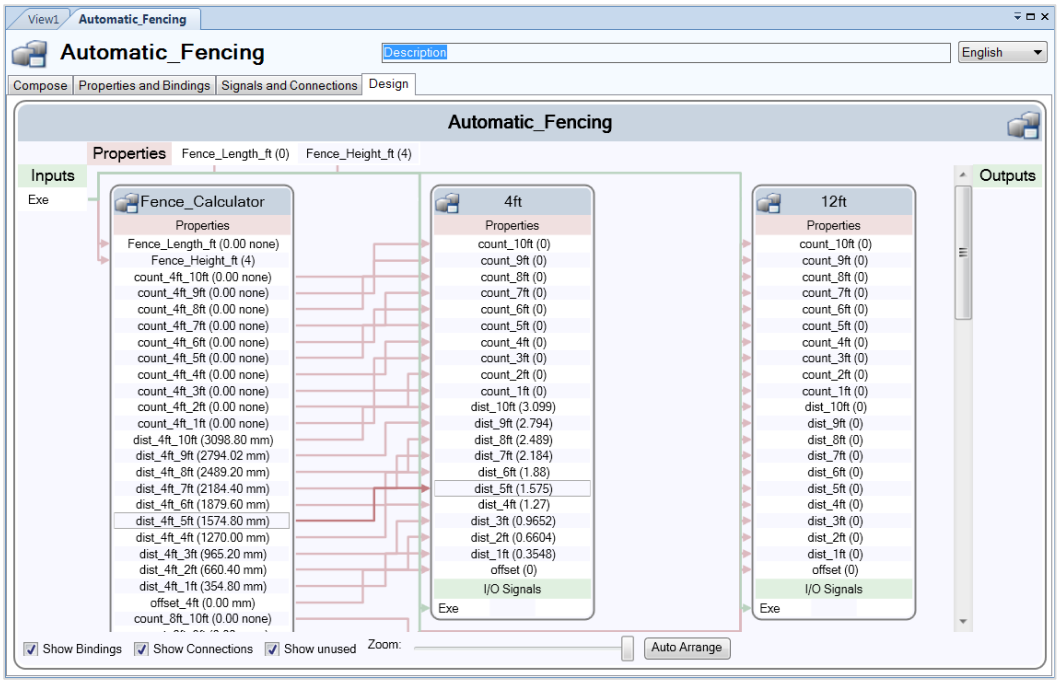


Figure 110: Maintenance Fencing Design

5. The sub components 4ft, 8ft, 10ft, and 12ft are all alike except they rely on different imported geometry and inputs. For this reason only the 4ft subcomponent will be discussed. This component has three main parts: the Matrix Repeater, the Hide Command, and the Expression. There is one of each for each segment of fence.

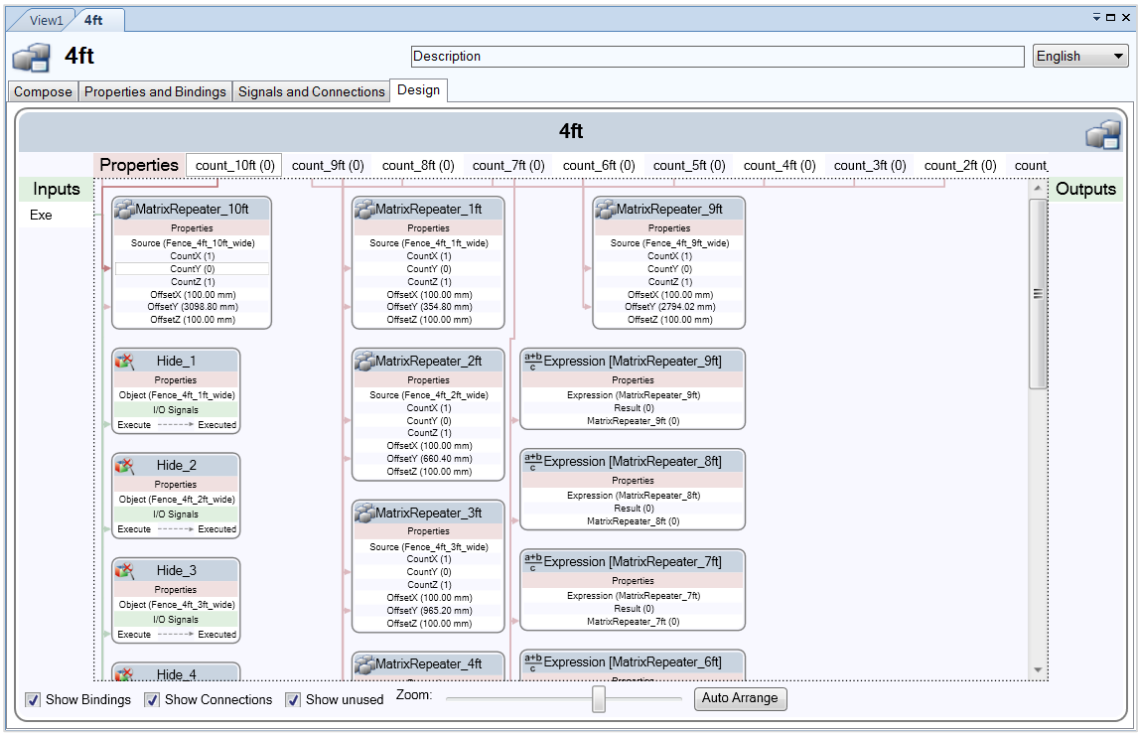


Figure 111: Maintenance Fencing Height Design

6. Matrix Repeater: This function is responsible for multiplying its source component linearly at a certain offset. It accepts two inputs, Count and Offset, which determine the number of times to multiply the source object and the distance between each multiplication.
7. Hide Command: This function is responsible for hiding the source object. This component relies on internal geometry and the internal geometry must be hidden when not in use. This takes one input: Execute.
8. Expression: This function is responsible for offsetting the matrix repeaters. It takes one input, offset, and uses this value do a linear transformation in the Y direction.

Automatic Track

The track Libraries that this component uses are self-contained; however, the carriage, gantry, and boom libraries are linked and need to be placed in their respective folders with their proper names.

Carriage:

S:\\Library Build\\Custom_Lib\\Carriage\\Carriage.rslib

Gantry:

S:\\Library Build\\Custom_Lib\\Gantry\\Gantry_2500H.rslib
S:\\Library Build\\Custom_Lib\\Gantry\\Gantry_3500H.rslib
S:\\Library Build\\Custom_Lib\\Gantry\\Gantry_4000H.rslib
S:\\Library Build\\Custom_Lib\\Gantry\\Inv_Gantry_2500H.rslib
S:\\Library Build\\Custom_Lib\\Gantry\\Inv_Gantry_3500H.rslib
S:\\Library Build\\Custom_Lib\\Gantry\\Inv_Gantry_4000H.rslib

Boom:

S:\\Library Build\\Custom_Lib\\Gantry\\Boom\\Boom_1500.rslib
S:\\Library Build\\Custom_Lib\\Gantry\\Boom\\Boom_2000.rslib
S:\\Library Build\\Custom_Lib\\Gantry\\Boom\\Boom_2500.rslib
S:\\Library Build\\Custom_Lib\\Gantry\\Boom\\Boom_3000.rslib

1. To access the smart component Libraries, right click on the smart component in the Layout tab and select *Disconnect Library*.

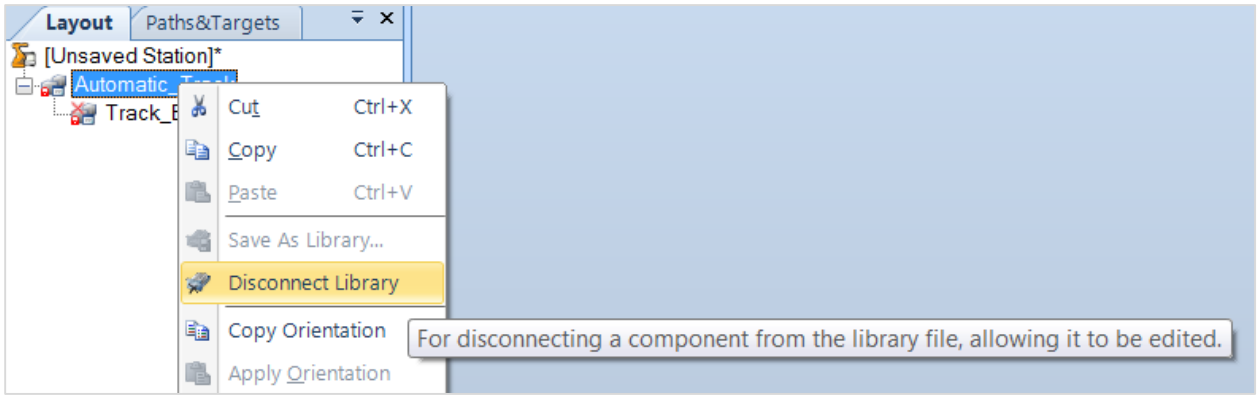


Figure 112: Maintenance Disconnect Track Library

2. Once the library has been disconnected again, right click on the smart component and select *Edit Component*.

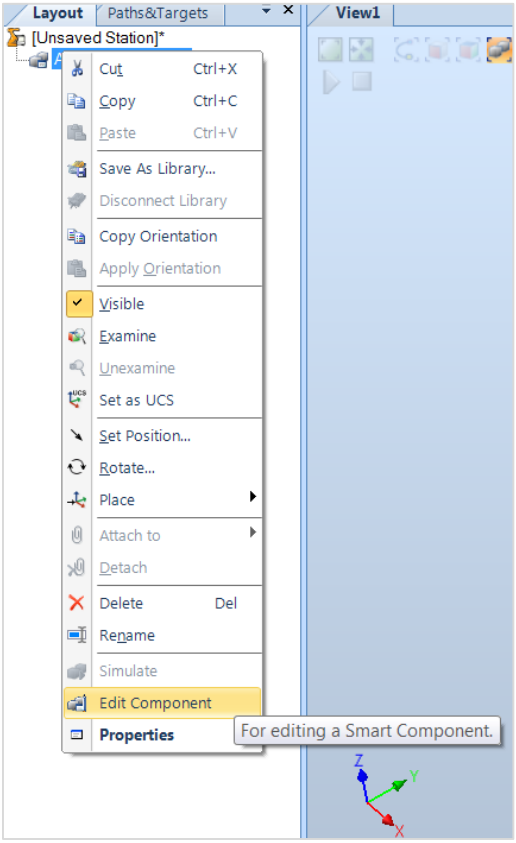


Figure 113: Maintenance Edit Track Component

3. This will open all of the graphic programming. For modularity this program has been separated into five sub components. Components can be selected by double-clicking the desired component.

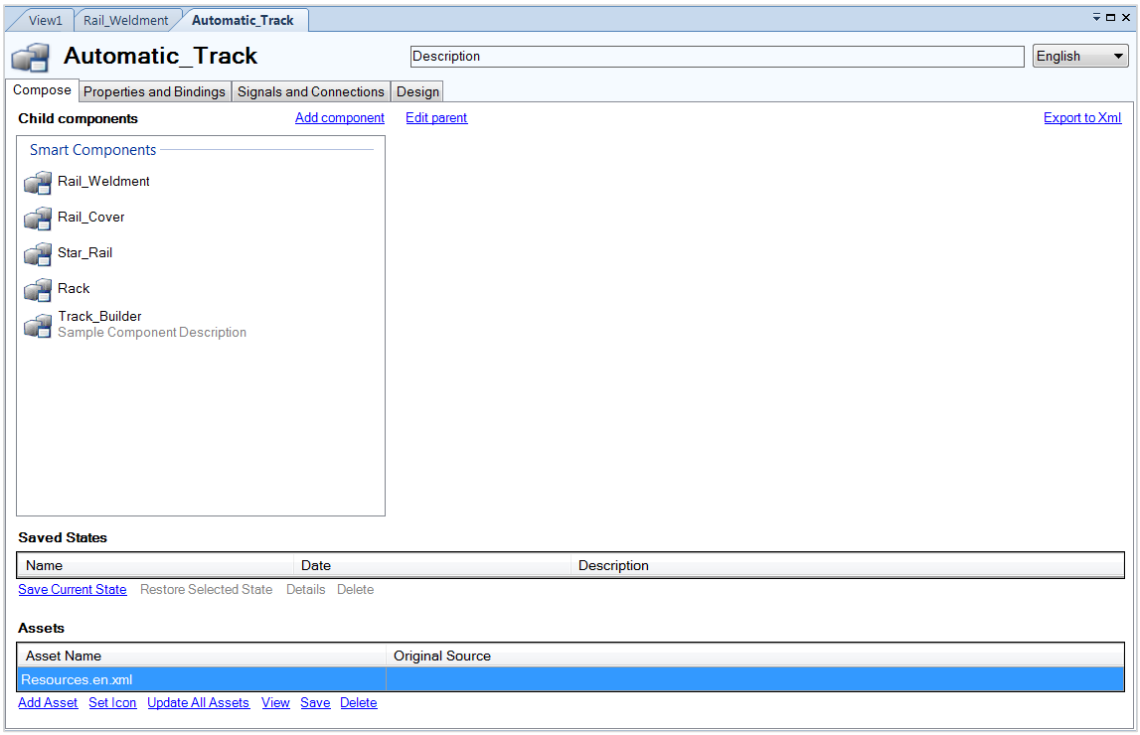


Figure 114: Maintenance Track Composition

4. Track_Builder determines the number of each section (Rail_Weldment, Rail_Cover, Star_Rail, and Rack) needed to build the track length specified. This component then outputs the count, distance, and offset that will be used by the other components. More about this component can be found in the Programming of Track_Builder.

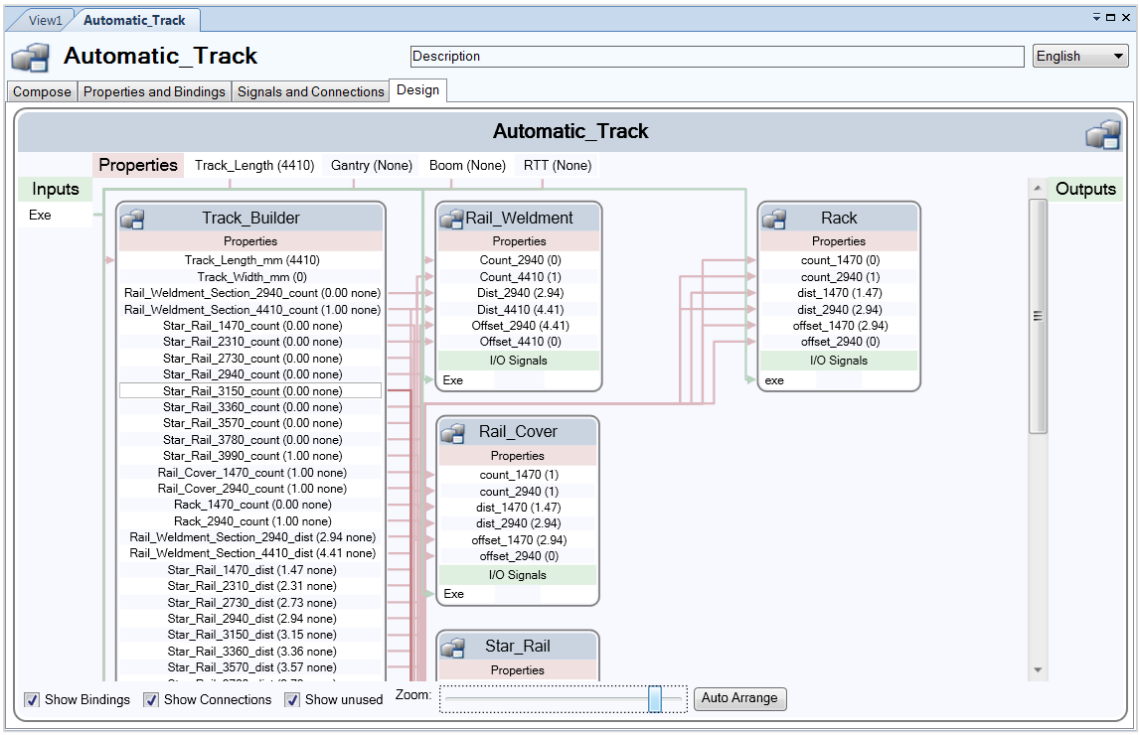


Figure 115: Maintenance Track Design

5. The sub components Rail_Weldment, Rail_Cover, Star_Rail, and Rack all function on similar principles except they rely on different imported geometry and inputs. For this reason only the Rail_Weldment subcomponent will be discussed. This component has three main parts: the Matrix Repeater, the Hide Command, and the Expression. There is one of each for each segment of fence.

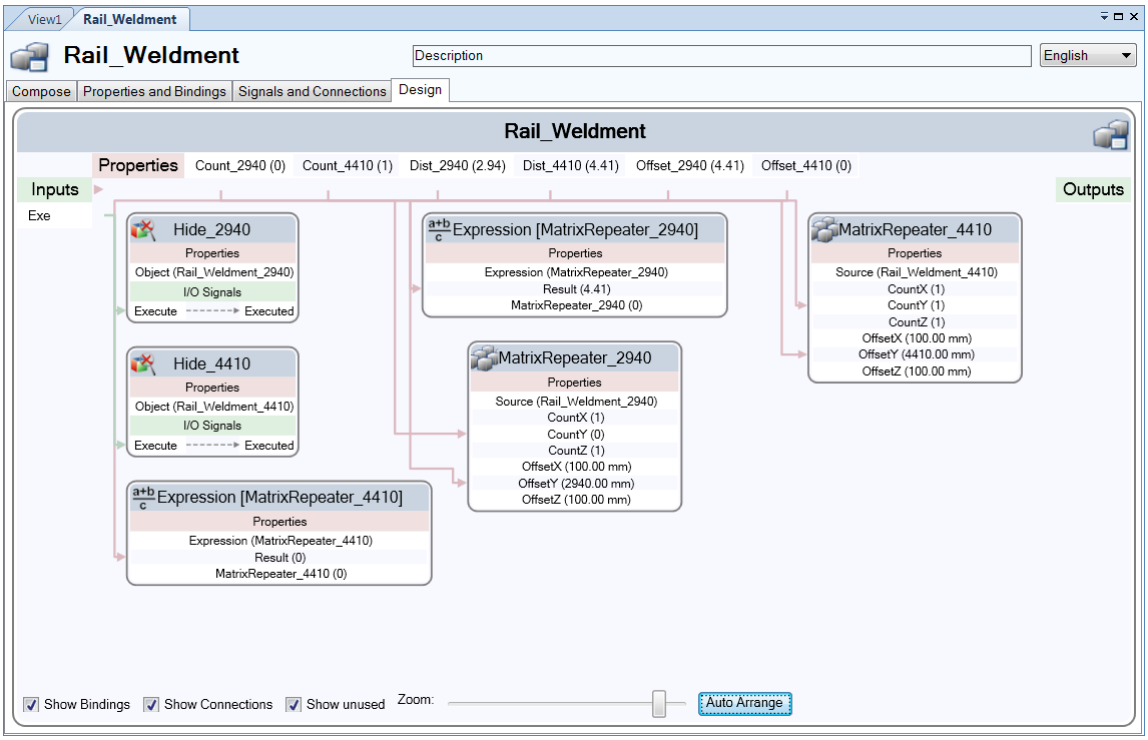


Figure 116: Maintenance Weldment Design

6. Matrix Repeater: This function is responsible for multiplying its source component linearly at a certain offset. It accepts two inputs, Count and Offset, which determine the number of times to multiply the source object and the distance between each multiplication.
7. Hide Command: This function is responsible for hiding the source object. This component relies on internal geometry and the internal geometry must be hidden when not in use. This takes one input, Execute.
8. Expression: This function is responsible for offsetting the matrix repeaters. This takes one input, offset, and uses this value do a linear transformation in the Y direction.

Automatic Light Curtain

The track Libraries that this component uses are linked and need to be placed in their respective folders with their proper names.

Light Curtains:

S:\\Library Build\\Custom_Lib\\Light_Curtain\\WolfLightCurtain_8ft.rslib
S:\\Library Build\\Custom_Lib\\Light_Curtain\\WolfLightCurtain_10ft.rslib
S:\\Library Build\\Custom_Lib\\Light_Curtain\\WolfLightCurtain_12ft.rslib

1. To access the smart component Libraries right click on the smart component in the Layout tab and select *Disconnect Library*.

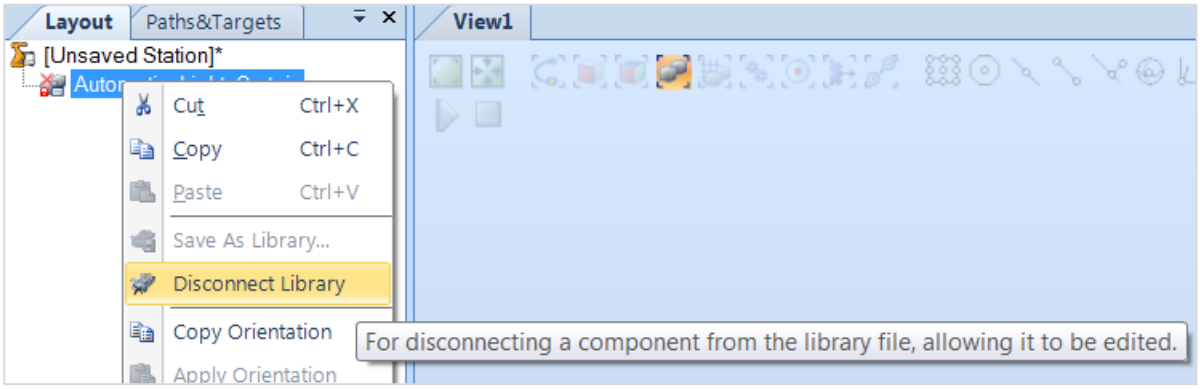


Figure 117: Maintenance Disconnect Light Curtain Library

2. Once the library has been disconnected again, right click on the smart component and select *Edit Component*.

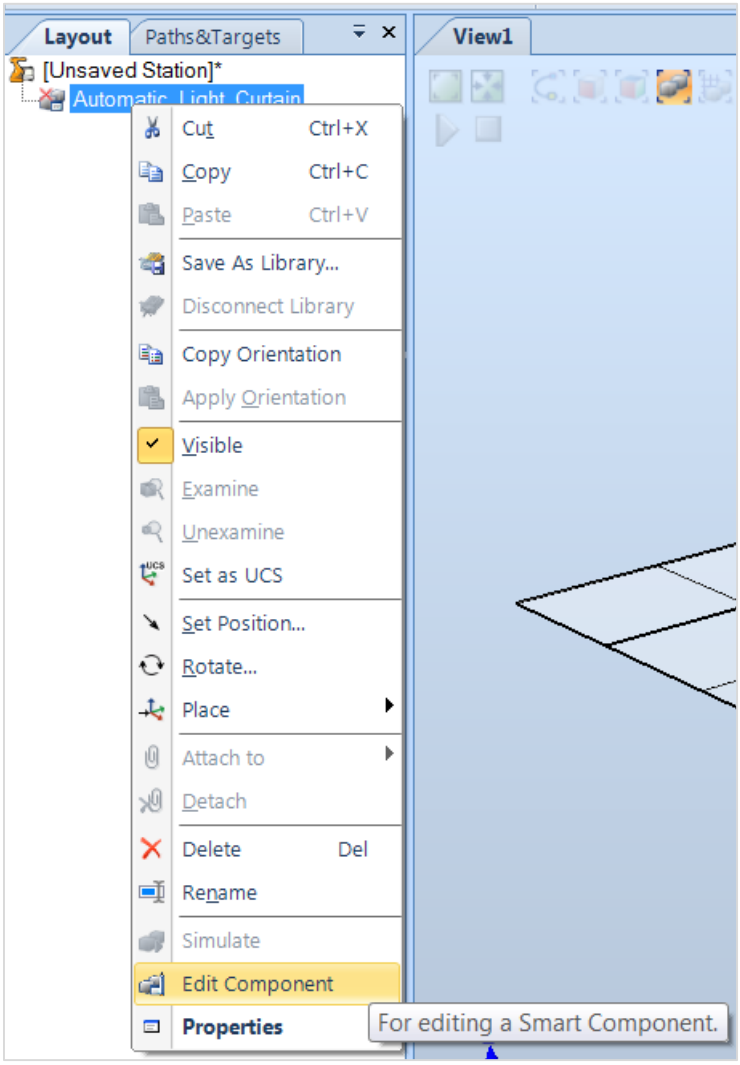


Figure 118: Maintenance Edit Light Curtain Component

3. This will open the graphic programming. This program relies on the smart component Safety and is manly a front end GUI for the user.

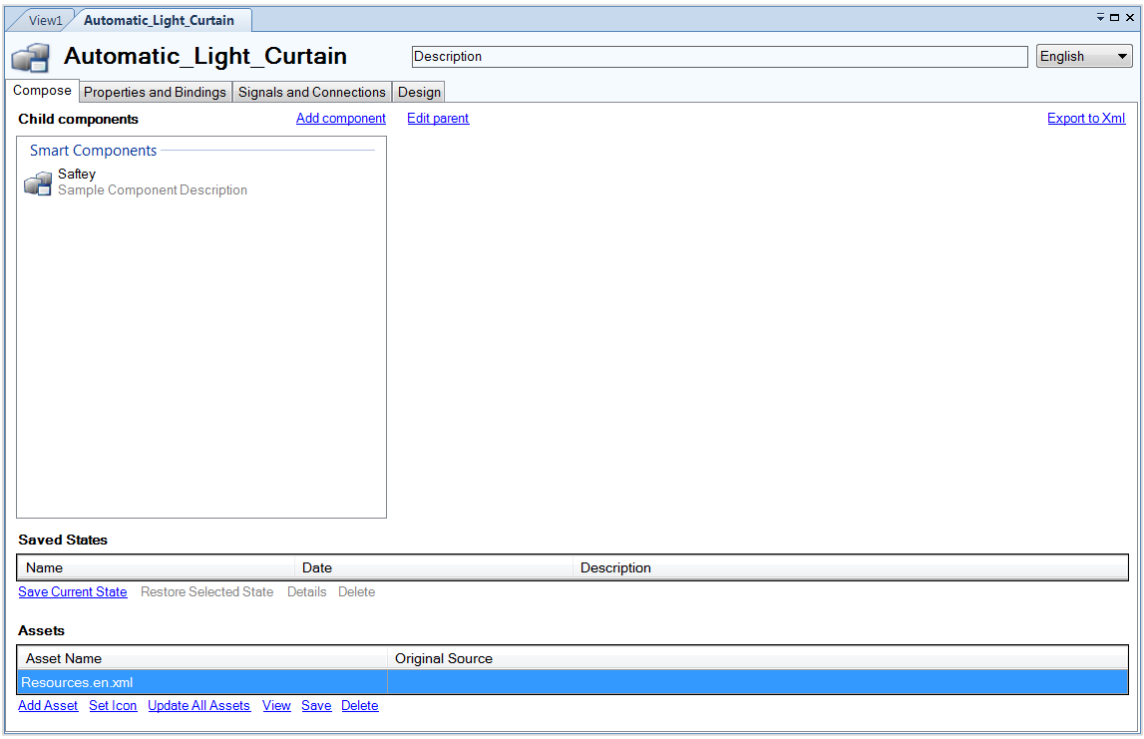


Figure 119: Maintenance Light Curtain Composition

4. Safety looks at all of the Light Curtains built and uses relative transformations to build the correct size laser curtain. More about this component can be found in the Programming of Safety.

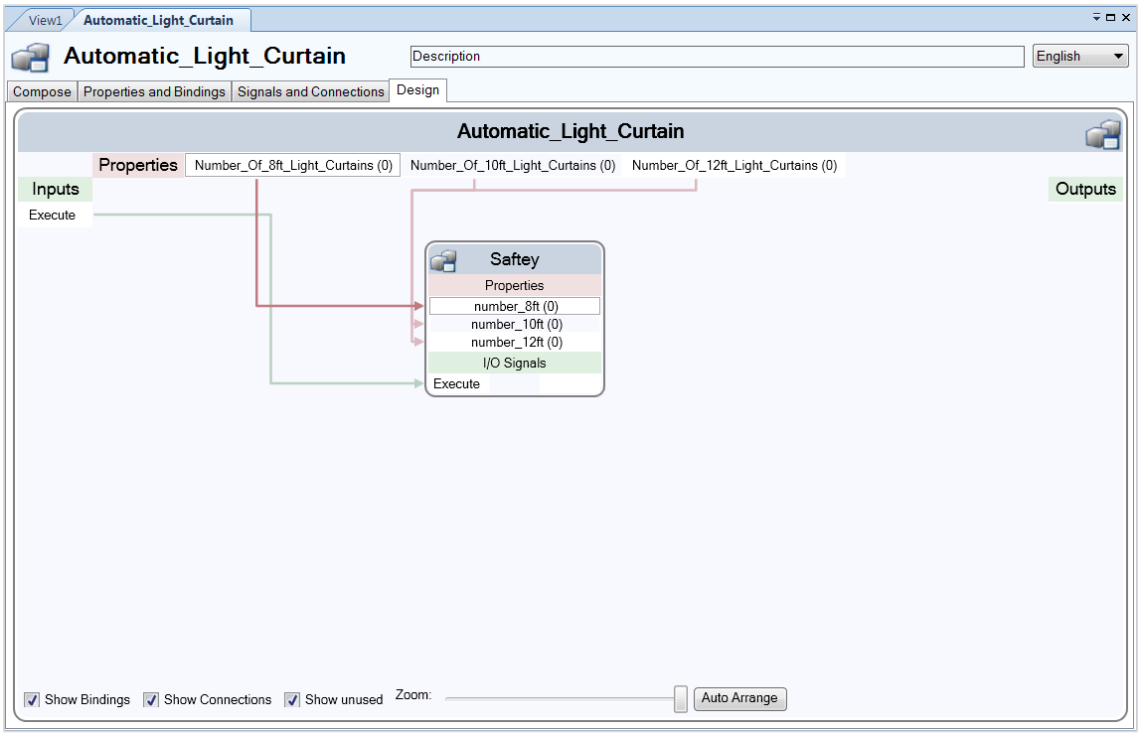
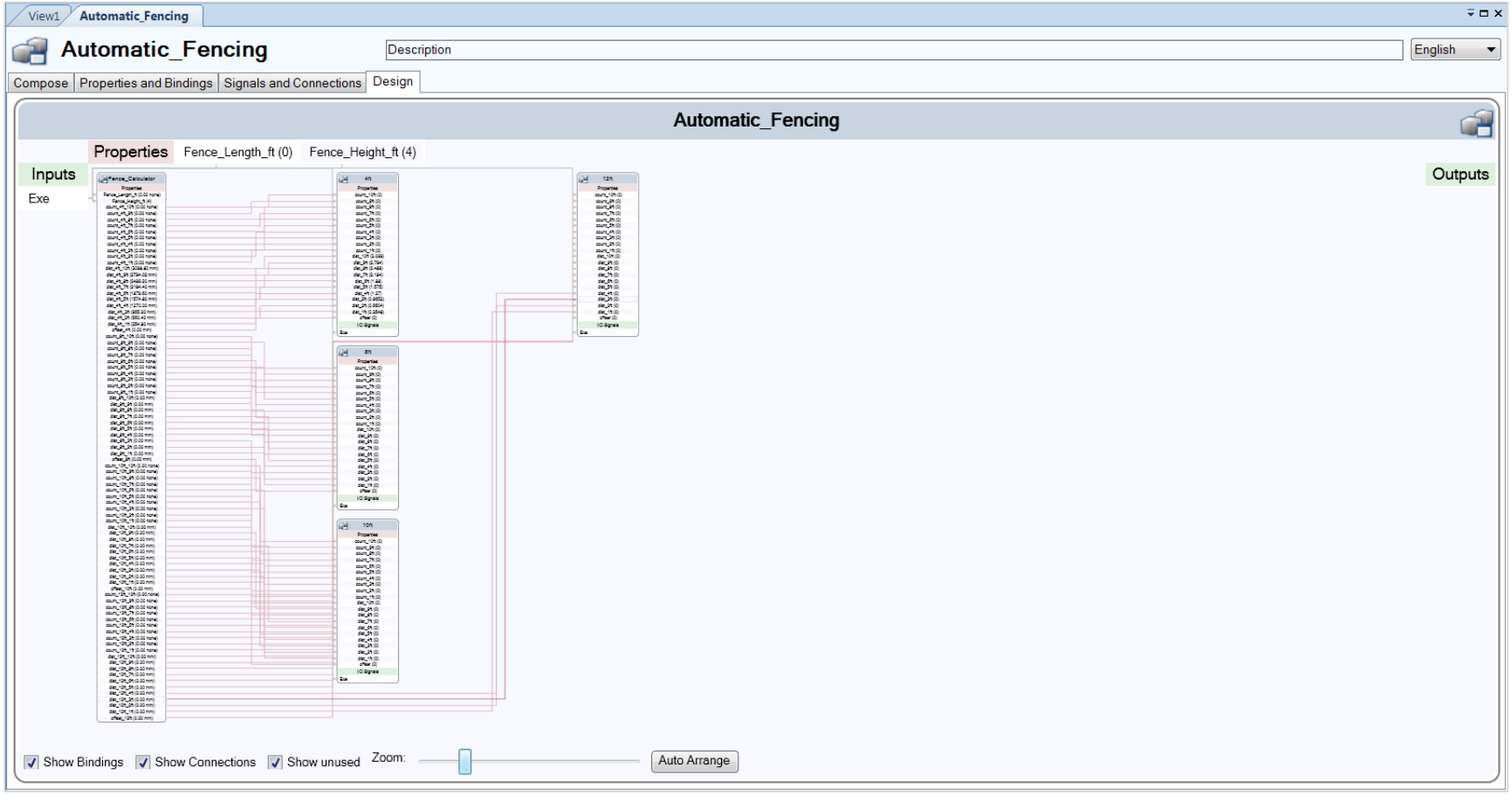


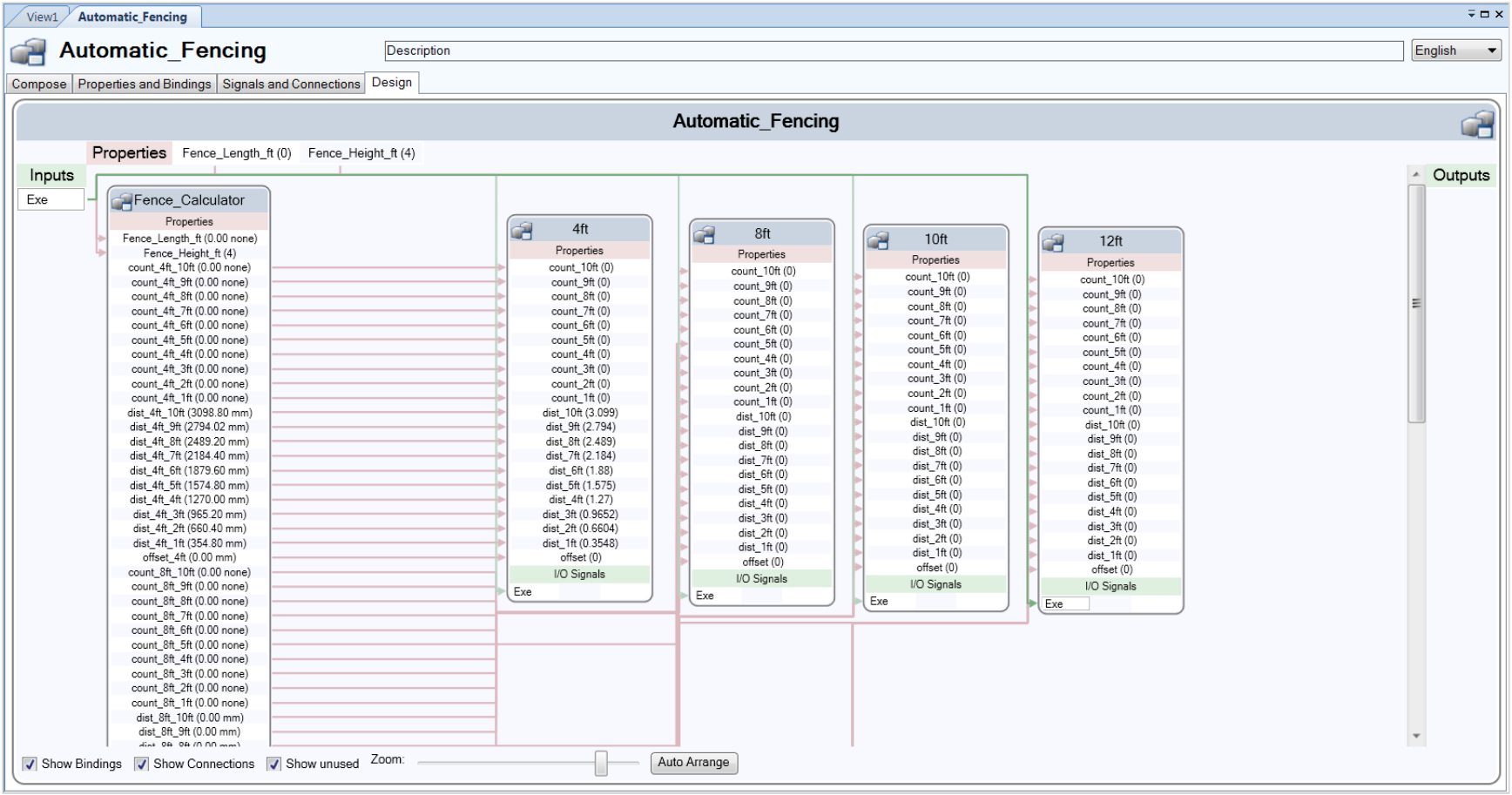
Figure 120: Maintenance Light Curtain Design

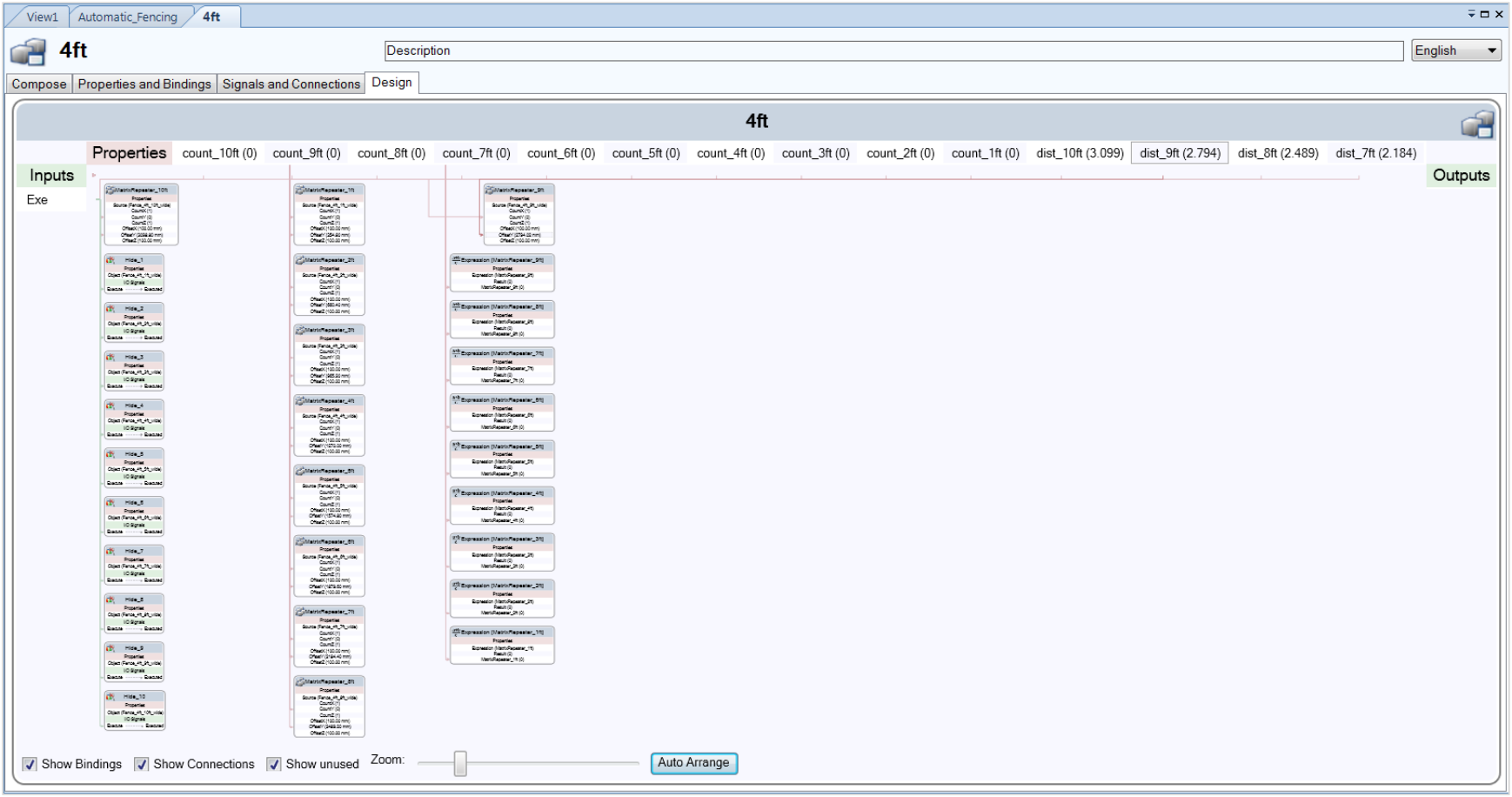
APPENDIX C. SOURCE CODE SMART COMPONENTS

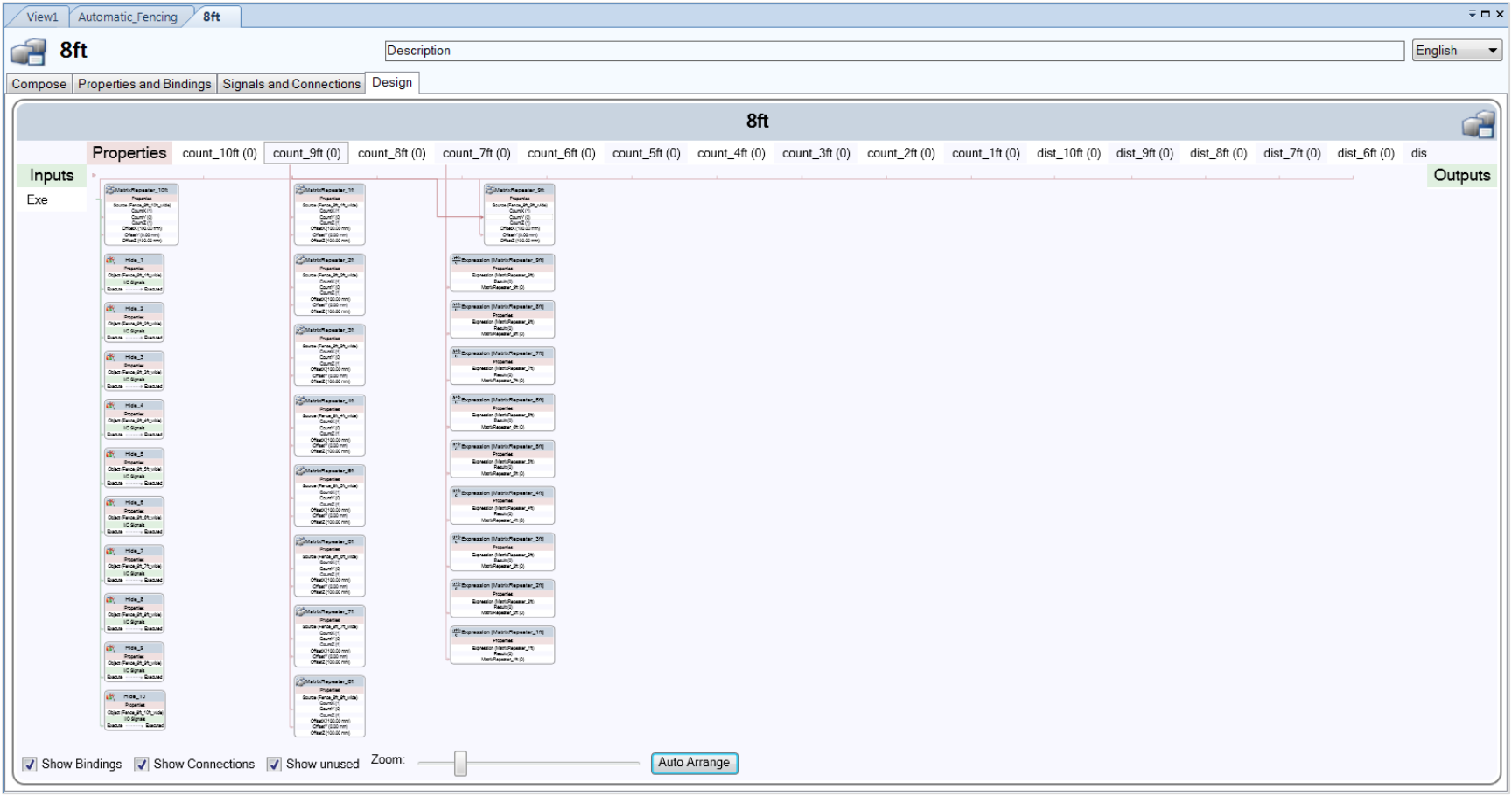
Fence_Calculator

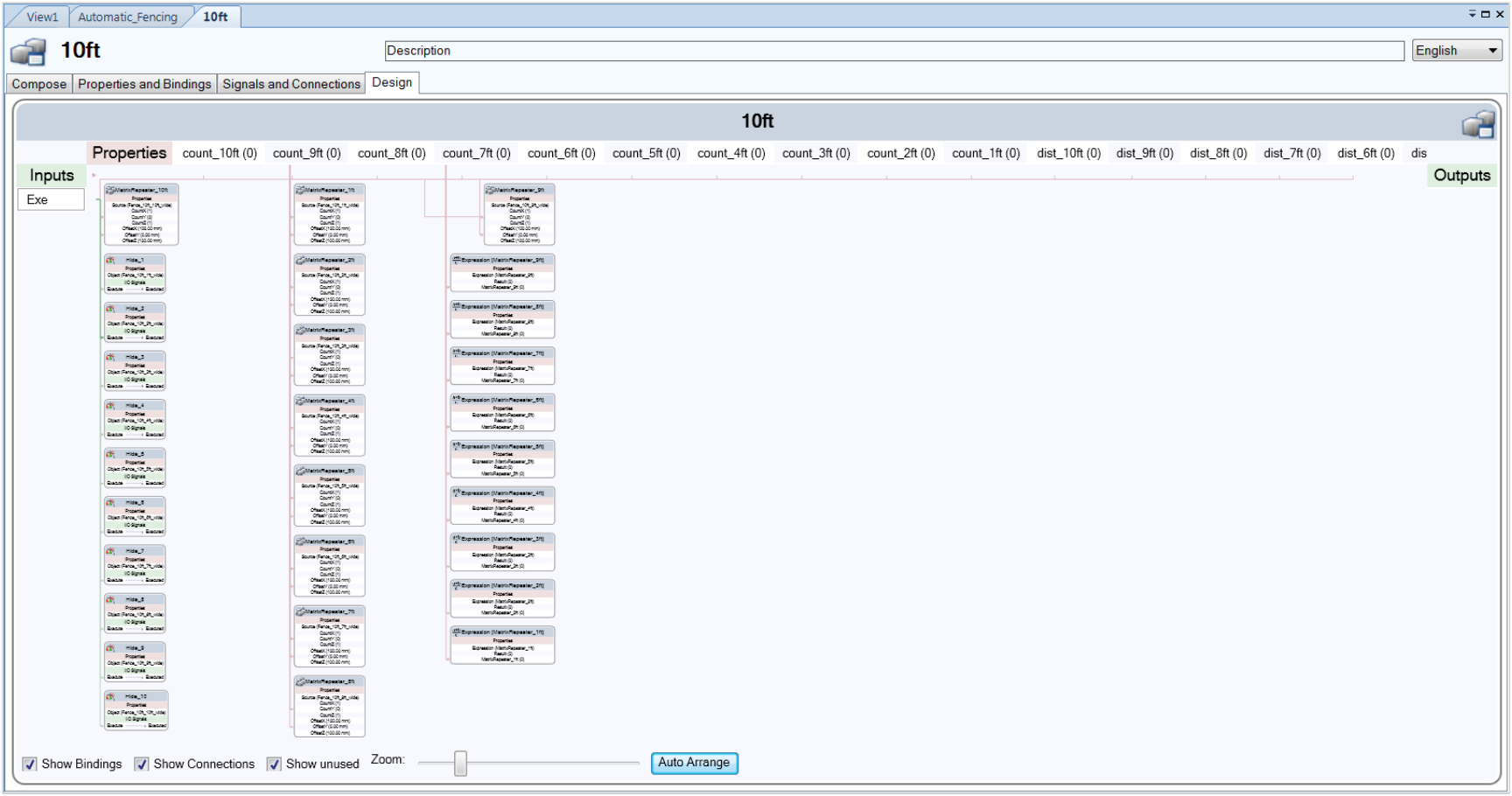
RobotStudio

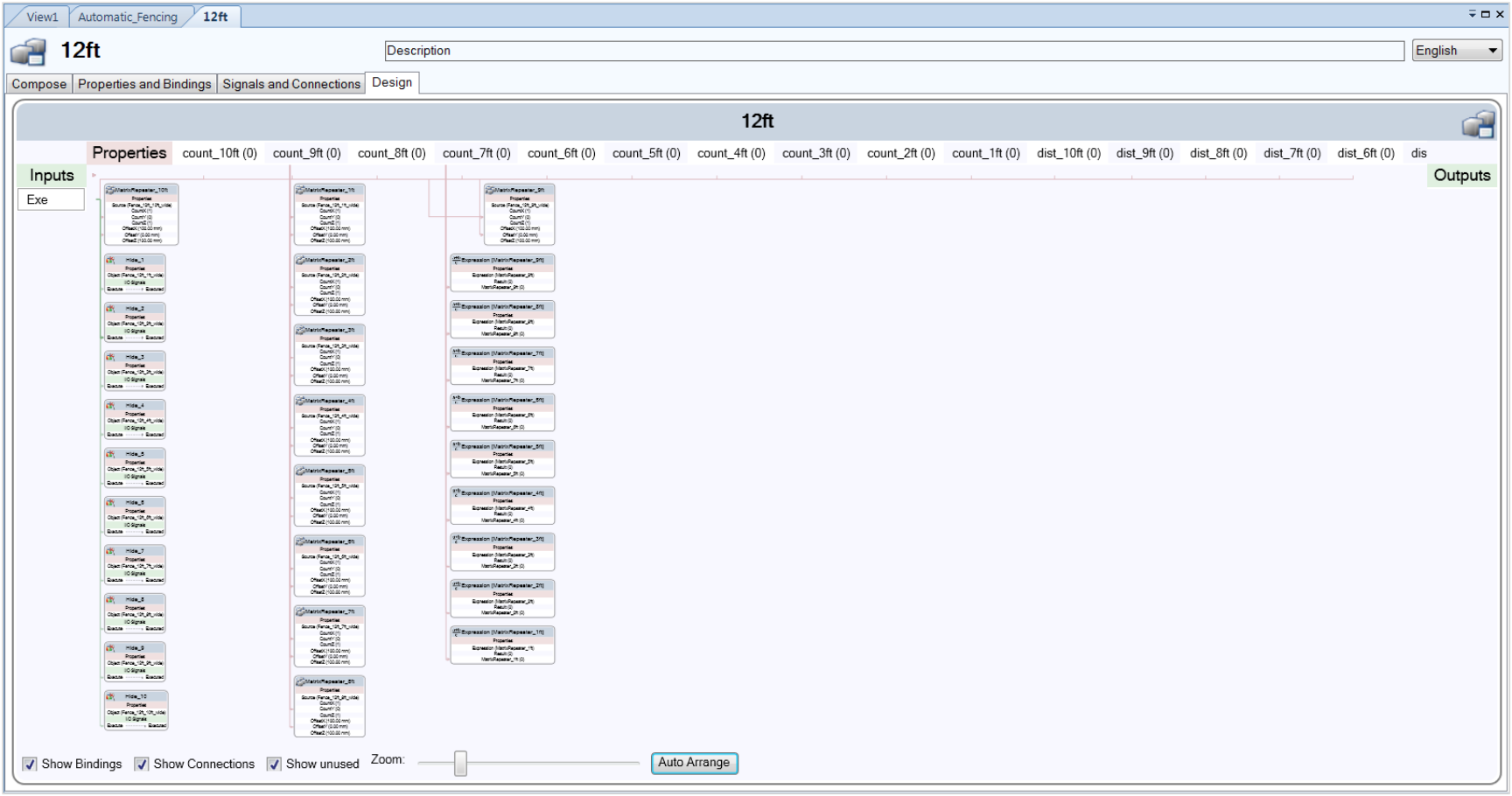



View1
Automatic_Fencing
Automatic_Fencing
Description
English
Compose
Properties and Bindings
Signals and Connections
Design
Automatic_Fencing
Properties
Fence_Length_ft (0)
Fence_Height_ft (4)
Inputs
Exe
Outputs
Fence_Calculator
Properties
Fence_Length_ft (0.00 none)
Fence_Height_ft (4)
count_4ft_10ft (0.00 none)
count_4ft_9ft (0.00 none)
count_4ft_8ft (0.00 none)
count_4ft_7ft (0.00 none)
count_4ft_6ft (0.00 none)
count_4ft_5ft (0.00 none)
count_4ft_4ft (0.00 none)
count_4ft_3ft (0.00 none)
count_4ft_2ft (0.00 none)
count_4ft_1ft (0.00 none)
dist_4ft_10ft (3098.80 mm)
dist_4ft_9ft (2794.02 mm)
dist_4ft_8ft (2489.20 mm)
dist_4ft_7ft (2184.40 mm)
dist_4ft_6ft (1879.60 mm)
dist_4ft_5ft (1574.80 mm)
dist_4ft_4ft (1270.00 mm)
dist_4ft_3ft (965.20 mm)
dist_4ft_2ft (660.40 mm)
dist_4ft_1ft (354.80 mm)
offset_4ft (0.00 mm)
count_8ft_10ft (0.00 none)
count_8ft_9ft (0.00 none)
count_8ft_8ft (0.00 none)
count_8ft_7ft (0.00 none)
count_8ft_6ft (0.00 none)
count_8ft_5ft (0.00 none)
count_8ft_4ft (0.00 none)
count_8ft_3ft (0.00 none)
count_8ft_2ft (0.00 none)
count_8ft_1ft (0.00 none)
dist_8ft_10ft (0.00 mm)
dist_8ft_9ft (0.00 mm)
4ft
Properties
count_10ft (0)
count_9ft (0)
count_8ft (0)
count_7ft (0)
count_6ft (0)
count_5ft (0)
count_4ft (0)
count_3ft (0)
count_2ft (0)
count_1ft (0)
dist_10ft (3.099)
dist_9ft (2.794)
dist_8ft (2.489)
dist_7ft (2.184)
dist_6ft (1.88)
dist_5ft (1.575)
dist_4ft (1.27)
dist_3ft (0.9652)
dist_2ft (0.6604)
dist_1ft (0.3548)
offset (0)
I/O Signals
Exe
8ft
Properties
count_10ft (0)
count_9ft (0)
count_8ft (0)
count_7ft (0)
count_6ft (0)
count_5ft (0)
count_4ft (0)
count_3ft (0)
count_2ft (0)
count_1ft (0)
dist_10ft (0)
dist_9ft (0)
dist_8ft (0)
dist_7ft (0)
dist_6ft (0)
dist_5ft (0)
dist_4ft (0)
dist_3ft (0)
dist_2ft (0)
dist_1ft (0)
offset (0)
I/O Signals
Exe
10ft
Properties
count_10ft (0)
count_9ft (0)
count_8ft (0)
count_7ft (0)
count_6ft (0)
count_5ft (0)
count_4ft (0)
count_3ft (0)
count_2ft (0)
count_1ft (0)
dist_10ft (0)
dist_9ft (0)
dist_8ft (0)
dist_7ft (0)
dist_6ft (0)
dist_5ft (0)
dist_4ft (0)
dist_3ft (0)
dist_2ft (0)
dist_1ft (0)
offset (0)
I/O Signals
Exe
12ft
Properties
count_10ft (0)
count_9ft (0)
count_8ft (0)
count_7ft (0)
count_6ft (0)
count_5ft (0)
count_4ft (0)
count_3ft (0)
count_2ft (0)
count_1ft (0)
dist_10ft (0)
dist_9ft (0)
dist_8ft (0)
dist_7ft (0)
dist_6ft (0)
dist_5ft (0)
dist_4ft (0)
dist_3ft (0)
dist_2ft (0)
dist_1ft (0)
offset (0)
I/O Signals
Exe
Show Bindings
Show Connections
Show unused
Zoom:
Auto Arrange


View1
Automatic_Fencing
4ft
4ft
Description
English
Compose
Properties and Bindings
Signals and Connections
Design
4ft
Properties
count_10ft (0)
count_9ft (0)
count_8ft (0)
count_7ft (0)
count_6ft (0)
count_5ft (0)
count_4ft (0)
count_3ft (0)
count_2ft (0)
count_1ft (0)
dist_10ft (3.099)
dist_9ft (2.794)
dist_8ft (2.489)
dist_7ft (2.184)
Inputs
Exe
Outputs
Show Bindings
Show Connections
Show unused
Zoom:
Auto Arrange


View1
Automatic_Fencing
8ft
8ft
Description
English
Compose
Properties and Bindings
Signals and Connections
Design
8ft
Properties
count_10ft (0)
count_9ft (0)
count_8ft (0)
count_7ft (0)
count_6ft (0)
count_5ft (0)
count_4ft (0)
count_3ft (0)
count_2ft (0)
count_1ft (0)
dist_10ft (0)
dist_9ft (0)
dist_8ft (0)
dist_7ft (0)
dist_6ft (0)
dis
Inputs
Exe
Outputs
Show Bindings
Show Connections
Show unused
Zoom:
Auto Arrange


View1
Automatic_Fencing
10ft
10ft
Description
English
Compose
Properties and Bindings
Signals and Connections
Design
10ft
Properties
count_10ft (0)
count_9ft (0)
count_8ft (0)
count_7ft (0)
count_6ft (0)
count_5ft (0)
count_4ft (0)
count_3ft (0)
count_2ft (0)
count_1ft (0)
dist_10ft (0)
dist_9ft (0)
dist_8ft (0)
dist_7ft (0)
dist_6ft (0)
dis
Inputs
Exe
Outputs
Show Bindings
Show Connections
Show unused
Zoom:
Auto Arrange


View1
Automatic_Fencing
12ft
12ft
Description
English
Compose
Properties and Bindings
Signals and Connections
Design
12ft
Properties
count_10ft (0)
count_9ft (0)
count_8ft (0)
count_7ft (0)
count_6ft (0)
count_5ft (0)
count_4ft (0)
count_3ft (0)
count_2ft (0)
count_1ft (0)
dist_10ft (0)
dist_9ft (0)
dist_8ft (0)
dist_7ft (0)
dist_6ft (0)
dis
Inputs
Exe
Outputs
Show Bindings
Show Connections
Show unused
Zoom:
Auto Arrange

CodeBehind

```
using System;
using System.Collections.Generic;
using System.Text;

using ABB.Robotics.Math;
using ABB.Robotics.RobotStudio;
using ABB.Robotics.RobotStudio.Environment;
using ABB.Robotics.RobotStudio.Stations;
using ABB.Robotics.RobotStudio.Stations.Forms;

namespace Fence_Calculator
{
   public class CodeBehind : SmartComponentCodeBehind
   {
      public override void OnPropertyValueChanged(SmartComponent component, DynamicProperty
changedProperty, object oldValue)
      {
         int x; x = 9;
         double[] fence;
         fence = new double[10];
         double temp; temp = 0;


         bool name = !(changedProperty.Name == "Fence_Length_ft");
         bool name2 = !(changedProperty.Name == "Fence_Height_ft");

         //Check if value has changed
         if (!name | !name2 )
         {
            //Get Robot Studio Variables
            double Orig_fencelength = (double)component.Properties["Fence_Length_ft"].Value;
            int Orig_Fence_Height = (int)component.Properties["Fence_Height_ft"].Value;

            //Calculate number of each fence segment
            fence[9] = Orig_fencelength / 10;
            fence[9] = (int)Math.Floor(fence[9]);
            temp = Orig_fencelength % 10;
            x--;

            while (x >= 0)
            {
               fence[x] = temp / (x + 1);
               fence[x] = (int)Math.Floor(fence[x]);
               temp = temp % (x + 1);
               x--;
            }
```

```
//Based on the number of fence segments and the fence height to determine the offset and output all
variables            if (Orig_Fence_Height == 4)
        {
            component.Properties["count_4ft_10ft"].Value = (fence[9]);
            component.Properties["count_4ft_9ft"].Value = (fence[8]);
            component.Properties["count_4ft_8ft"].Value = (fence[7]);
            component.Properties["count_4ft_7ft"].Value = (fence[6]);
            component.Properties["count_4ft_6ft"].Value = (fence[5]);
            component.Properties["count_4ft_5ft"].Value = (fence[4]);
            component.Properties["count_4ft_4ft"].Value = (fence[3]);
            component.Properties["count_4ft_3ft"].Value = (fence[2]);
            component.Properties["count_4ft_2ft"].Value = (fence[1]);
            component.Properties["count_4ft_1ft"].Value = (fence[0]);
            component.Properties["dist_4ft_10ft"].Value = (3.0988);
            component.Properties["dist_4ft_9ft"].Value = (2.79402);
            component.Properties["dist_4ft_8ft"].Value = (2.4892);
            component.Properties["dist_4ft_7ft"].Value = (2.1844);
            component.Properties["dist_4ft_6ft"].Value = (1.8796);
            component.Properties["dist_4ft_5ft"].Value = (1.5748);
            component.Properties["dist_4ft_4ft"].Value = (1.27);
            component.Properties["dist_4ft_3ft"].Value = (.9652);
            component.Properties["dist_4ft_2ft"].Value = (.6604);
            component.Properties["dist_4ft_1ft"].Value = (.3548);
            component.Properties["offset_4ft"].Value = (fence[9] * 3.0988);

            //Clear and previous values that may have been used
            component.Properties["count_8ft_10ft"].Value = (0);
            component.Properties["count_8ft_9ft"].Value = (0);
            component.Properties["count_8ft_8ft"].Value = (0);
            component.Properties["count_8ft_7ft"].Value = (0);
            component.Properties["count_8ft_6ft"].Value = (0);
            component.Properties["count_8ft_5ft"].Value = (0);
            component.Properties["count_8ft_4ft"].Value = (0);
            component.Properties["count_8ft_3ft"].Value = (0);
            component.Properties["count_8ft_2ft"].Value = (0);
            component.Properties["count_8ft_1ft"].Value = (0);
            component.Properties["dist_8ft_10ft"].Value = (0);
            component.Properties["dist_8ft_9ft"].Value = (0);
            component.Properties["dist_8ft_8ft"].Value = (0);
            component.Properties["dist_8ft_7ft"].Value = (0);
            component.Properties["dist_8ft_6ft"].Value = (0);
            component.Properties["dist_8ft_5ft"].Value = (0);
            component.Properties["dist_8ft_4ft"].Value = (0);
            component.Properties["dist_8ft_3ft"].Value = (0);
            component.Properties["dist_8ft_2ft"].Value = (0);
            component.Properties["dist_8ft_1ft"].Value = (0);
            component.Properties["offset_8ft"].Value = (0);
```

```
    component.Properties["count_10ft_10ft"].Value = (0);
    component.Properties["count_10ft_9ft"].Value = (0);
    component.Properties["count_10ft_8ft"].Value = (0);
    component.Properties["count_10ft_7ft"].Value = (0);
    component.Properties["count_10ft_6ft"].Value = (0);
    component.Properties["count_10ft_5ft"].Value = (0);
    component.Properties["count_10ft_4ft"].Value = (0);
    component.Properties["count_10ft_3ft"].Value = (0);
    component.Properties["count_10ft_2ft"].Value = (0);
    component.Properties["count_10ft_1ft"].Value = (0);
    component.Properties["dist_10ft_10ft"].Value = (0);
    component.Properties["dist_10ft_9ft"].Value = (0);
    component.Properties["dist_10ft_8ft"].Value = (0);
    component.Properties["dist_10ft_7ft"].Value = (0);
    component.Properties["dist_10ft_6ft"].Value = (0);
    component.Properties["dist_10ft_5ft"].Value = (0);
    component.Properties["dist_10ft_4ft"].Value = (0);
    component.Properties["dist_10ft_3ft"].Value = (0);
    component.Properties["dist_10ft_2ft"].Value = (0);
    component.Properties["dist_10ft_1ft"].Value = (0);
    component.Properties["offset_10ft"].Value = (0);

    component.Properties["count_12ft_10ft"].Value = (0);
    component.Properties["count_12ft_9ft"].Value = (0);
    component.Properties["count_12ft_8ft"].Value = (0);
    component.Properties["count_12ft_7ft"].Value = (0);
    component.Properties["count_12ft_6ft"].Value = (0);
    component.Properties["count_12ft_5ft"].Value = (0);
    component.Properties["count_12ft_4ft"].Value = (0);
    component.Properties["count_12ft_3ft"].Value = (0);
    component.Properties["count_12ft_2ft"].Value = (0);
    component.Properties["count_12ft_1ft"].Value = (0);
    component.Properties["dist_12ft_10ft"].Value = (0);
    component.Properties["dist_12ft_9ft"].Value = (0);
    component.Properties["dist_12ft_8ft"].Value = (0);
    component.Properties["dist_12ft_7ft"].Value = (0);
    component.Properties["dist_12ft_6ft"].Value = (0);
    component.Properties["dist_12ft_5ft"].Value = (0);
    component.Properties["dist_12ft_4ft"].Value = (0);
    component.Properties["dist_12ft_3ft"].Value = (0);
    component.Properties["dist_12ft_2ft"].Value = (0);
    component.Properties["dist_12ft_1ft"].Value = (0);
    component.Properties["offset_12ft"].Value = (0);
}

if (Orig_Fence_Height == 8)
{
```

```
component.Properties[“count_8ft_10ft”].Value = (fence[9]);
component.Properties[“count_8ft_9ft”].Value = (fence[8]);
component.Properties[“count_8ft_8ft”].Value = (fence[7]);
component.Properties[“count_8ft_7ft”].Value = (fence[6]);
component.Properties[“count_8ft_6ft”].Value = (fence[5]);
component.Properties[“count_8ft_5ft”].Value = (fence[4]);
component.Properties[“count_8ft_4ft”].Value = (fence[3]);
component.Properties[“count_8ft_3ft”].Value = (fence[2]);
component.Properties[“count_8ft_2ft”].Value = (fence[1]);
component.Properties[“count_8ft_1ft”].Value = (fence[0]);
component.Properties[“dist_8ft_10ft”].Value = (3.0988);
component.Properties[“dist_8ft_9ft”].Value = (2.79402);
component.Properties[“dist_8ft_8ft”].Value = (2.4892);
component.Properties[“dist_8ft_7ft”].Value = (2.1844);
component.Properties[“dist_8ft_6ft”].Value = (1.8796);
component.Properties[“dist_8ft_5ft”].Value = (1.5748);
component.Properties[“dist_8ft_4ft”].Value = (1.27);
component.Properties[“dist_8ft_3ft”].Value = (.9652);
component.Properties[“dist_8ft_2ft”].Value = (.6604);
component.Properties[“dist_8ft_1ft”].Value = (.3548);
component.Properties[“offset_8ft”].Value = (fence[9] * 3.0988);

//Clear and previous values that may have been used
component.Properties[“count_4ft_10ft”].Value = (0);
component.Properties[“count_4ft_9ft”].Value = (0);
component.Properties[“count_4ft_8ft”].Value = (0);
component.Properties[“count_4ft_7ft”].Value = (0);
component.Properties[“count_4ft_6ft”].Value = (0);
component.Properties[“count_4ft_5ft”].Value = (0);
component.Properties[“count_4ft_4ft”].Value = (0);
component.Properties[“count_4ft_3ft”].Value = (0);
component.Properties[“count_4ft_2ft”].Value = (0);
component.Properties[“count_4ft_1ft”].Value = (0);
component.Properties[“dist_4ft_10ft”].Value = (0);
component.Properties[“dist_4ft_9ft”].Value = (0);
component.Properties[“dist_4ft_8ft”].Value = (0);
component.Properties[“dist_4ft_7ft”].Value = (0);
component.Properties[“dist_4ft_6ft”].Value = (0);
component.Properties[“dist_4ft_5ft”].Value = (0);
component.Properties[“dist_4ft_4ft”].Value = (0);
component.Properties[“dist_4ft_3ft”].Value = (0);
component.Properties[“dist_4ft_2ft”].Value = (0);
component.Properties[“dist_4ft_1ft”].Value = (0);
component.Properties[“offset_4ft”].Value = (0);

component.Properties[“count_10ft_10ft”].Value = (0);
component.Properties[“count_10ft_9ft”].Value = (0);
component.Properties[“count_10ft_8ft”].Value = (0);
```

```
    component.Properties["count_10ft_7ft"].Value = (0);
    component.Properties["count_10ft_6ft"].Value = (0);
    component.Properties["count_10ft_5ft"].Value = (0);
    component.Properties["count_10ft_4ft"].Value = (0);
    component.Properties["count_10ft_3ft"].Value = (0);
    component.Properties["count_10ft_2ft"].Value = (0);
    component.Properties["count_10ft_1ft"].Value = (0);
    component.Properties["dist_10ft_10ft"].Value = (0);
    component.Properties["dist_10ft_9ft"].Value = (0);
    component.Properties["dist_10ft_8ft"].Value = (0);
    component.Properties["dist_10ft_7ft"].Value = (0);
    component.Properties["dist_10ft_6ft"].Value = (0);
    component.Properties["dist_10ft_5ft"].Value = (0);
    component.Properties["dist_10ft_4ft"].Value = (0);
    component.Properties["dist_10ft_3ft"].Value = (0);
    component.Properties["dist_10ft_2ft"].Value = (0);
    component.Properties["dist_10ft_1ft"].Value = (0);
    component.Properties["offset_10ft"].Value = (0);

    component.Properties["count_12ft_10ft"].Value = (0);
    component.Properties["count_12ft_9ft"].Value = (0);
    component.Properties["count_12ft_8ft"].Value = (0);
    component.Properties["count_12ft_7ft"].Value = (0);
    component.Properties["count_12ft_6ft"].Value = (0);
    component.Properties["count_12ft_5ft"].Value = (0);
    component.Properties["count_12ft_4ft"].Value = (0);
    component.Properties["count_12ft_3ft"].Value = (0);
    component.Properties["count_12ft_2ft"].Value = (0);
    component.Properties["count_12ft_1ft"].Value = (0);
    component.Properties["dist_12ft_10ft"].Value = (0);
    component.Properties["dist_12ft_9ft"].Value = (0);
    component.Properties["dist_12ft_8ft"].Value = (0);
    component.Properties["dist_12ft_7ft"].Value = (0);
    component.Properties["dist_12ft_6ft"].Value = (0);
    component.Properties["dist_12ft_5ft"].Value = (0);
    component.Properties["dist_12ft_4ft"].Value = (0);
    component.Properties["dist_12ft_3ft"].Value = (0);
    component.Properties["dist_12ft_2ft"].Value = (0);
    component.Properties["dist_12ft_1ft"].Value = (0);
    component.Properties["offset_12ft"].Value = (0);
}

if (Orig_Fence_Height == 10)
{
    component.Properties["count_10ft_10ft"].Value = (fence[9]);
    component.Properties["count_10ft_9ft"].Value = (fence[8]);
    component.Properties["count_10ft_8ft"].Value = (fence[7]);
    component.Properties["count_10ft_7ft"].Value = (fence[6]);
```

```
component.Properties[“count_10ft_6ft”].Value = (fence[5]);
component.Properties[“count_10ft_5ft”].Value = (fence[4]);
component.Properties[“count_10ft_4ft”].Value = (fence[3]);
component.Properties[“count_10ft_3ft”].Value = (fence[2]);
component.Properties[“count_10ft_2ft”].Value = (fence[1]);
component.Properties[“count_10ft_1ft”].Value = (fence[0]);
component.Properties[“dist_10ft_10ft”].Value = (3.0988);
component.Properties[“dist_10ft_9ft”].Value = (2.79402);
component.Properties[“dist_10ft_8ft”].Value = (2.4892);
component.Properties[“dist_10ft_7ft”].Value = (2.1844);
component.Properties[“dist_10ft_6ft”].Value = (1.8796);
component.Properties[“dist_10ft_5ft”].Value = (1.5748);
component.Properties[“dist_10ft_4ft”].Value = (1.27);
component.Properties[“dist_10ft_3ft”].Value = (.9652);
component.Properties[“dist_10ft_2ft”].Value = (.6604);
component.Properties[“dist_10ft_1ft”].Value = (.3548);
component.Properties[“offset_10ft”].Value = (fence[9] * 3.0988);

//Clear and previous values that may have been used
component.Properties[“count_4ft_10ft”].Value = (0);
component.Properties[“count_4ft_9ft”].Value = (0);
component.Properties[“count_4ft_8ft”].Value = (0);
component.Properties[“count_4ft_7ft”].Value = (0);
component.Properties[“count_4ft_6ft”].Value = (0);
component.Properties[“count_4ft_5ft”].Value = (0);
component.Properties[“count_4ft_4ft”].Value = (0);
component.Properties[“count_4ft_3ft”].Value = (0);
component.Properties[“count_4ft_2ft”].Value = (0);
component.Properties[“count_4ft_1ft”].Value = (0);
component.Properties[“dist_4ft_10ft”].Value = (0);
component.Properties[“dist_4ft_9ft”].Value = (0);
component.Properties[“dist_4ft_8ft”].Value = (0);
component.Properties[“dist_4ft_7ft”].Value = (0);
component.Properties[“dist_4ft_6ft”].Value = (0);
component.Properties[“dist_4ft_5ft”].Value = (0);
component.Properties[“dist_4ft_4ft”].Value = (0);
component.Properties[“dist_4ft_3ft”].Value = (0);
component.Properties[“dist_4ft_2ft”].Value = (0);
component.Properties[“dist_4ft_1ft”].Value = (0);
component.Properties[“offset_4ft”].Value = (0);


component.Properties[“count_8ft_10ft”].Value = (0);
component.Properties[“count_8ft_9ft”].Value = (0);
component.Properties[“count_8ft_8ft”].Value = (0);
component.Properties[“count_8ft_7ft”].Value = (0);
component.Properties[“count_8ft_6ft”].Value = (0);
component.Properties[“count_8ft_5ft”].Value = (0);
```

```
    component.Properties[“count_8ft_4ft”].Value = (0);
    component.Properties[“count_8ft_3ft”].Value = (0);
    component.Properties[“count_8ft_2ft”].Value = (0);
    component.Properties[“count_8ft_1ft”].Value = (0);
    component.Properties[“dist_8ft_10ft”].Value = (0);
    component.Properties[“dist_8ft_9ft”].Value = (0);
    component.Properties[“dist_8ft_8ft”].Value = (0);
    component.Properties[“dist_8ft_7ft”].Value = (0);
    component.Properties[“dist_8ft_6ft”].Value = (0);
    component.Properties[“dist_8ft_5ft”].Value = (0);
    component.Properties[“dist_8ft_4ft”].Value = (0);
    component.Properties[“dist_8ft_3ft”].Value = (0);
    component.Properties[“dist_8ft_2ft”].Value = (0);
    component.Properties[“dist_8ft_1ft”].Value = (0);
    component.Properties[“offset_8ft”].Value = (0);

    component.Properties[“count_12ft_10ft”].Value = (0);
    component.Properties[“count_12ft_9ft”].Value = (0);
    component.Properties[“count_12ft_8ft”].Value = (0);
    component.Properties[“count_12ft_7ft”].Value = (0);
    component.Properties[“count_12ft_6ft”].Value = (0);
    component.Properties[“count_12ft_5ft”].Value = (0);
    component.Properties[“count_12ft_4ft”].Value = (0);
    component.Properties[“count_12ft_3ft”].Value = (0);
    component.Properties[“count_12ft_2ft”].Value = (0);
    component.Properties[“count_12ft_1ft”].Value = (0);
    component.Properties[“dist_12ft_10ft”].Value = (0);
    component.Properties[“dist_12ft_9ft”].Value = (0);
    component.Properties[“dist_12ft_8ft”].Value = (0);
    component.Properties[“dist_12ft_7ft”].Value = (0);
    component.Properties[“dist_12ft_6ft”].Value = (0);
    component.Properties[“dist_12ft_5ft”].Value = (0);
    component.Properties[“dist_12ft_4ft”].Value = (0);
    component.Properties[“dist_12ft_3ft”].Value = (0);
    component.Properties[“dist_12ft_2ft”].Value = (0);
    component.Properties[“dist_12ft_1ft”].Value = (0);
    component.Properties[“offset_12ft”].Value = (0);
}

if (Orig_Fence_Height == 12)
{
    component.Properties[“count_12ft_10ft”].Value = (fence[9]);
    component.Properties[“count_12ft_9ft”].Value = (fence[8]);
    component.Properties[“count_12ft_8ft”].Value = (fence[7]);
    component.Properties[“count_12ft_7ft”].Value = (fence[6]);
    component.Properties[“count_12ft_6ft”].Value = (fence[5]);
    component.Properties[“count_12ft_5ft”].Value = (fence[4]);
    component.Properties[“count_12ft_4ft”].Value = (fence[3]);
```

```
component.Properties["count_12ft_3ft"].Value = (fence[2]);
component.Properties["count_12ft_2ft"].Value = (fence[1]);
component.Properties["count_12ft_1ft"].Value = (fence[0]);
component.Properties["dist_12ft_10ft"].Value = (3.0988);
component.Properties["dist_12ft_9ft"].Value = (2.79402);
component.Properties["dist_12ft_8ft"].Value = (2.4892);
component.Properties["dist_12ft_7ft"].Value = (2.1844);
component.Properties["dist_12ft_6ft"].Value = (1.8796);
component.Properties["dist_12ft_5ft"].Value = (1.5748);
component.Properties["dist_12ft_4ft"].Value = (1.27);
component.Properties["dist_12ft_3ft"].Value = (.9652);
component.Properties["dist_12ft_2ft"].Value = (.6604);
component.Properties["dist_12ft_1ft"].Value = (.3548);
component.Properties["offset_12ft"].Value = (fence[9] * 3.0988);

//Clear and previous values that may have been used
component.Properties["count_4ft_10ft"].Value = (0);
component.Properties["count_4ft_9ft"].Value = (0);
component.Properties["count_4ft_8ft"].Value = (0);
component.Properties["count_4ft_7ft"].Value = (0);
component.Properties["count_4ft_6ft"].Value = (0);
component.Properties["count_4ft_5ft"].Value = (0);
component.Properties["count_4ft_4ft"].Value = (0);
component.Properties["count_4ft_3ft"].Value = (0);
component.Properties["count_4ft_2ft"].Value = (0);
component.Properties["count_4ft_1ft"].Value = (0);
component.Properties["dist_4ft_10ft"].Value = (0);
component.Properties["dist_4ft_9ft"].Value = (0);
component.Properties["dist_4ft_8ft"].Value = (0);
component.Properties["dist_4ft_7ft"].Value = (0);
component.Properties["dist_4ft_6ft"].Value = (0);
component.Properties["dist_4ft_5ft"].Value = (0);
component.Properties["dist_4ft_4ft"].Value = (0);
component.Properties["dist_4ft_3ft"].Value = (0);
component.Properties["dist_4ft_2ft"].Value = (0);
component.Properties["dist_4ft_1ft"].Value = (0);
component.Properties["offset_4ft"].Value = (0);


component.Properties["count_8ft_10ft"].Value = (0);
component.Properties["count_8ft_9ft"].Value = (0);
component.Properties["count_8ft_8ft"].Value = (0);
component.Properties["count_8ft_7ft"].Value = (0);
component.Properties["count_8ft_6ft"].Value = (0);
component.Properties["count_8ft_5ft"].Value = (0);
component.Properties["count_8ft_4ft"].Value = (0);
component.Properties["count_8ft_3ft"].Value = (0);
component.Properties["count_8ft_2ft"].Value = (0);
```

```
            component.Properties["count_8ft_1ft"].Value = (0);
            component.Properties["dist_8ft_10ft"].Value = (0);
            component.Properties["dist_8ft_9ft"].Value = (0);
            component.Properties["dist_8ft_8ft"].Value = (0);
            component.Properties["dist_8ft_7ft"].Value = (0);
            component.Properties["dist_8ft_6ft"].Value = (0);
            component.Properties["dist_8ft_5ft"].Value = (0);
            component.Properties["dist_8ft_4ft"].Value = (0);
            component.Properties["dist_8ft_3ft"].Value = (0);
            component.Properties["dist_8ft_2ft"].Value = (0);
            component.Properties["dist_8ft_1ft"].Value = (0);
            component.Properties["offset_8ft"].Value = (0);

            component.Properties["count_10ft_10ft"].Value = (0);
            component.Properties["count_10ft_9ft"].Value = (0);
            component.Properties["count_10ft_8ft"].Value = (0);
            component.Properties["count_10ft_7ft"].Value = (0);
            component.Properties["count_10ft_6ft"].Value = (0);
            component.Properties["count_10ft_5ft"].Value = (0);
            component.Properties["count_10ft_4ft"].Value = (0);
            component.Properties["count_10ft_3ft"].Value = (0);
            component.Properties["count_10ft_2ft"].Value = (0);
            component.Properties["count_10ft_1ft"].Value = (0);
            component.Properties["dist_10ft_10ft"].Value = (0);
            component.Properties["dist_10ft_9ft"].Value = (0);
            component.Properties["dist_10ft_8ft"].Value = (0);
            component.Properties["dist_10ft_7ft"].Value = (0);
            component.Properties["dist_10ft_6ft"].Value = (0);
            component.Properties["dist_10ft_5ft"].Value = (0);
            component.Properties["dist_10ft_4ft"].Value = (0);
            component.Properties["dist_10ft_3ft"].Value = (0);
            component.Properties["dist_10ft_2ft"].Value = (0);
            component.Properties["dist_10ft_1ft"].Value = (0);
            component.Properties["offset_10ft"].Value = (0);
        }
      }
    }
  }
}
```

XML

```
<?xml version=”1.0” encoding=”utf-8” ?>
<lc:LibraryCompiler xmlns:lc=”urn:abb-robotics-robotstudio-librarycompiler”
                    xmlns=”urn:abb-robotics-robotstudio-graphiccomponent”>
        <lc:Library fileName=”Fence_Calculator.rslib”>
                <lc:DocumentProperties>
                        <lc:Author>Joseph.Kopacz</lc:Author>
                        <lc:Image source=”Fence_Calculator.png”/>
                </lc:DocumentProperties>
                <SmartComponent name=”Fence_Calculator” icon=”Fence_Calculator.png”

        codeBehind=”Fence_Calculator.CodeBehind,Fence_Calculator.dll”
        canBeSimulated=”false”>
<Properties>

<!—This is a commented line built for the Input Variables
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx→
    <DynamicProperty name=”Fence_Length_ft” valueType=”System.Double” value=”0”>
     <Attribute key=”MinValue” value=”0”/>
     <Attribute key=”Quantity” value=”None”/>
    </DynamicProperty>

    <DynamicProperty name=”Fence_Height_ft” valueType=”System.Int32” value=”0”>
     <Attribute key=”Allowed” value=”4;8;10;12”/>
    </DynamicProperty>

    <!—This is a commented line built for the 4ft Height
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx→
    <DynamicProperty name=”count_4ft_10ft” valueType=”System.Double” value=”0”
readOnly=”true”>
     <Attribute key=”MinValue” value=”0”/>
     <Attribute key=”Quantity” value=”None”/>
    </DynamicProperty>
    <DynamicProperty name=”count_4ft_9ft” valueType=”System.Double” value=”0” readOnly=”true”>
     <Attribute key=”MinValue” value=”0”/>
     <Attribute key=”Quantity” value=”None”/>
    </DynamicProperty>
    <DynamicProperty name=”count_4ft_8ft” valueType=”System.Double” value=”0” readOnly=”true”>
     <Attribute key=”MinValue” value=”0”/>
     <Attribute key=”Quantity” value=”None”/>
    </DynamicProperty>
    <DynamicProperty name=”count_4ft_7ft” valueType=”System.Double” value=”0” readOnly=”true”>
     <Attribute key=”MinValue” value=”0”/>
     <Attribute key=”Quantity” value=”None”/>
    </DynamicProperty>
    <DynamicProperty name=”count_4ft_6ft” valueType=”System.Double” value=”0” readOnly=”true”>
     <Attribute key=”MinValue” value=”0”/>
     <Attribute key=”Quantity” value=”None”/>
```

```
</DynamicProperty>
<DynamicProperty name=”count_4ft_5ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
 <Attribute key=”Quantity” value=”None”/>
</DynamicProperty>
<DynamicProperty name=”count_4ft_4ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
 <Attribute key=”Quantity” value=”None”/>
</DynamicProperty>
<DynamicProperty name=”count_4ft_3ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
 <Attribute key=”Quantity” value=”None”/>
</DynamicProperty>
<DynamicProperty name=”count_4ft_2ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
 <Attribute key=”Quantity” value=”None”/>
</DynamicProperty>
<DynamicProperty name=”count_4ft_1ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
 <Attribute key=”Quantity” value=”None”/>
</DynamicProperty>

<DynamicProperty name=”dist_4ft_10ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
 <Attribute key=”Quantity” value=”Length”/>
</DynamicProperty>
<DynamicProperty name=”dist_4ft_9ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
 <Attribute key=”Quantity” value=”Length”/>
</DynamicProperty>
<DynamicProperty name=”dist_4ft_8ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
 <Attribute key=”Quantity” value=”Length”/>
</DynamicProperty>
<DynamicProperty name=”dist_4ft_7ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
 <Attribute key=”Quantity” value=”Length”/>
</DynamicProperty>
<DynamicProperty name=”dist_4ft_6ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
 <Attribute key=”Quantity” value=”Length”/>
</DynamicProperty>
<DynamicProperty name=”dist_4ft_5ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
 <Attribute key=”Quantity” value=”Length”/>
</DynamicProperty>
<DynamicProperty name=”dist_4ft_4ft” valueType=”System.Double” value=”0” readOnly=”true”>
 <Attribute key=”MinValue” value=”0”/>
```

```
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_4ft_3ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_4ft_2ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_4ft_1ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>

    <DynamicProperty name="offset_4ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>

    <!—This is a commented line built for the 8ft Height
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx→
    <DynamicProperty name="count_8ft_10ft" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_8ft_9ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_8ft_8ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_8ft_7ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_8ft_6ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_8ft_5ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_8ft_4ft" valueType="System.Double" value="0" readOnly="true">
```

```
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="None"/>
</DynamicProperty>
<DynamicProperty name="count_8ft_3ft" valueType="System.Double" value="0" readOnly="true">
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="None"/>
</DynamicProperty>
<DynamicProperty name="count_8ft_2ft" valueType="System.Double" value="0" readOnly="true">
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="None"/>
</DynamicProperty>
<DynamicProperty name="count_8ft_1ft" valueType="System.Double" value="0" readOnly="true">
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="None"/>
</DynamicProperty>

<DynamicProperty name="dist_8ft_10ft" valueType="System.Double" value="0" readOnly="true">
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="Length"/>
</DynamicProperty>
<DynamicProperty name="dist_8ft_9ft" valueType="System.Double" value="0" readOnly="true">
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="Length"/>
</DynamicProperty>
<DynamicProperty name="dist_8ft_8ft" valueType="System.Double" value="0" readOnly="true">
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="Length"/>
</DynamicProperty>
<DynamicProperty name="dist_8ft_7ft" valueType="System.Double" value="0" readOnly="true">
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="Length"/>
</DynamicProperty>
<DynamicProperty name="dist_8ft_6ft" valueType="System.Double" value="0" readOnly="true">
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="Length"/>
</DynamicProperty>
<DynamicProperty name="dist_8ft_5ft" valueType="System.Double" value="0" readOnly="true">
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="Length"/>
</DynamicProperty>
<DynamicProperty name="dist_8ft_4ft" valueType="System.Double" value="0" readOnly="true">
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="Length"/>
</DynamicProperty>
<DynamicProperty name="dist_8ft_3ft" valueType="System.Double" value="0" readOnly="true">
 <Attribute key="MinValue" value="0"/>
 <Attribute key="Quantity" value="Length"/>
</DynamicProperty>
```

```
    <DynamicProperty name="dist_8ft_2ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_8ft_1ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>

    <DynamicProperty name="offset_8ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>

    <!—This is a commented line built for the 10ft Height
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx→
    <DynamicProperty name="count_10ft_10ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_10ft_9ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_10ft_8ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_10ft_7ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_10ft_6ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_10ft_5ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_10ft_4ft" valueType="System.Double" value="0"
readOnly="true">
```

```
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_10ft_3ft" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_10ft_2ft" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_10ft_1ft" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <DynamicProperty name="dist_10ft_10ft" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_10ft_9ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_10ft_8ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_10ft_7ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_10ft_6ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_10ft_5ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_10ft_4ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
```

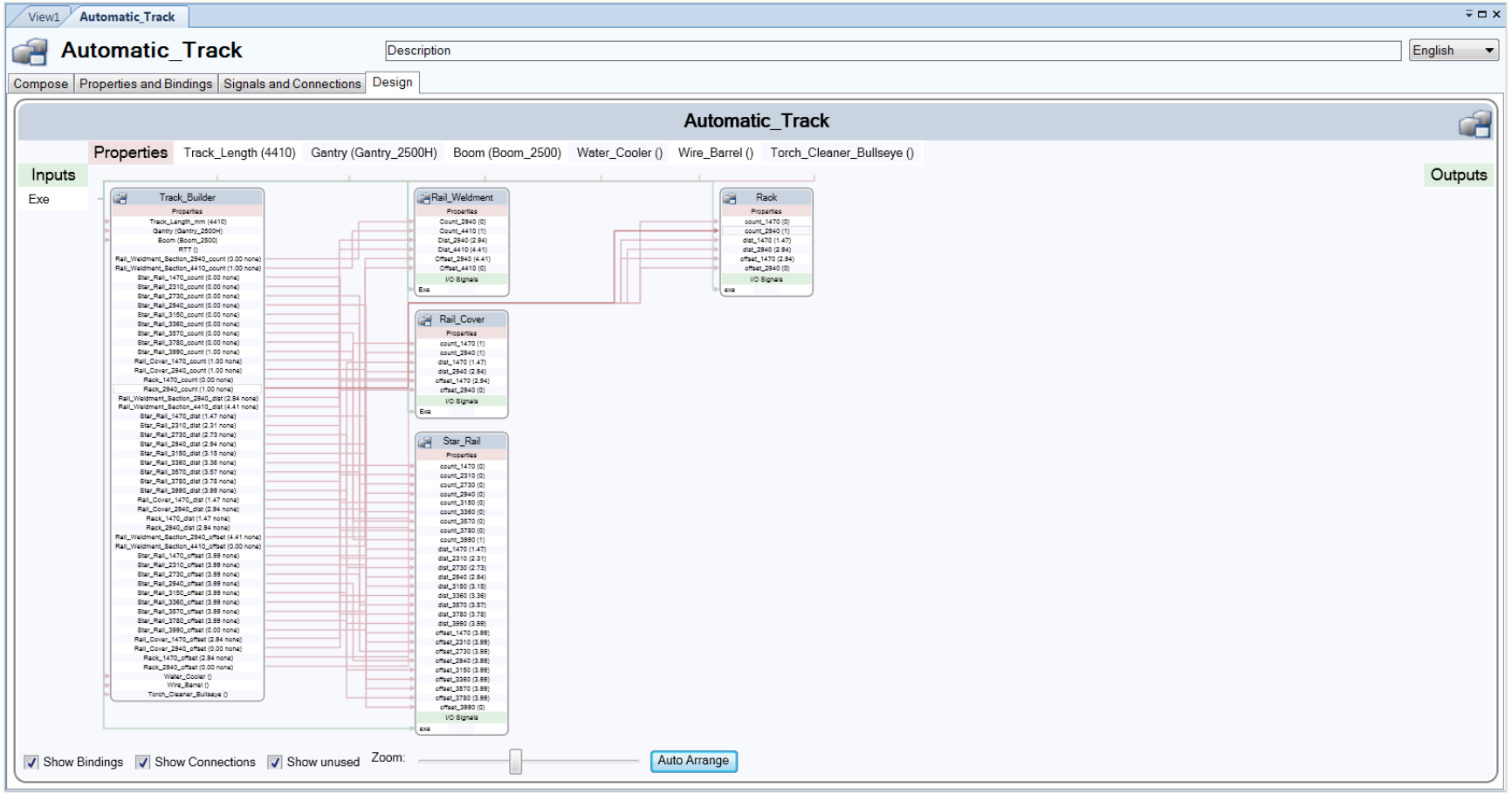
```
    <DynamicProperty name="dist_10ft_3ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_10ft_2ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_10ft_1ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>

    <DynamicProperty name="offset_10ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>

    <!—This is a commented line built for the 12ft Height
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx→
    <DynamicProperty name="count_12ft_10ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_12ft_9ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_12ft_8ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_12ft_7ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_12ft_6ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_12ft_5ft" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
```

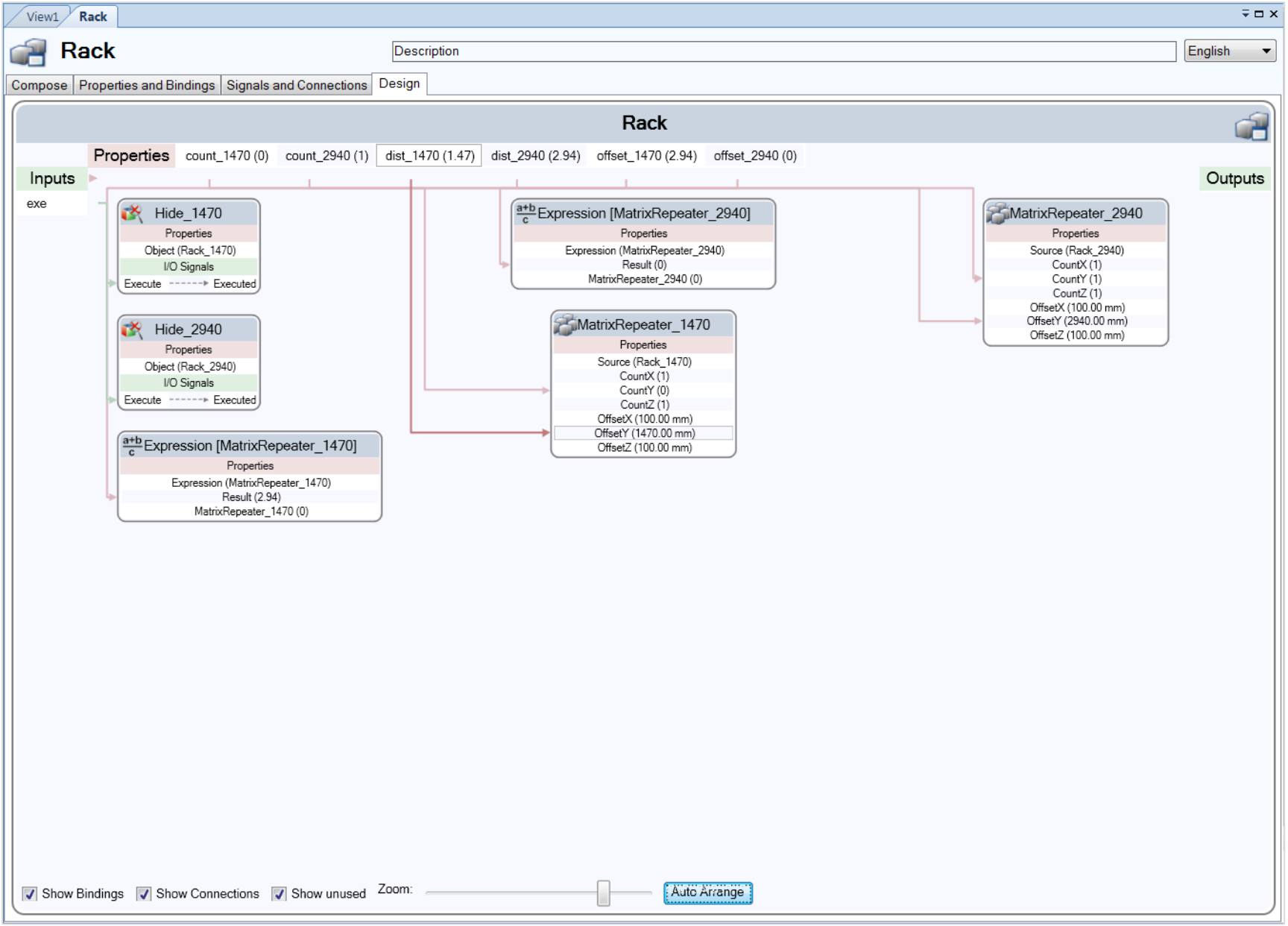
```
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_12ft_4ft" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_12ft_3ft" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_12ft_2ft" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="count_12ft_1ft" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <DynamicProperty name="dist_12ft_10ft" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_12ft_9ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_12ft_8ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_12ft_7ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_12ft_6ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_12ft_5ft" valueType="System.Double" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
```

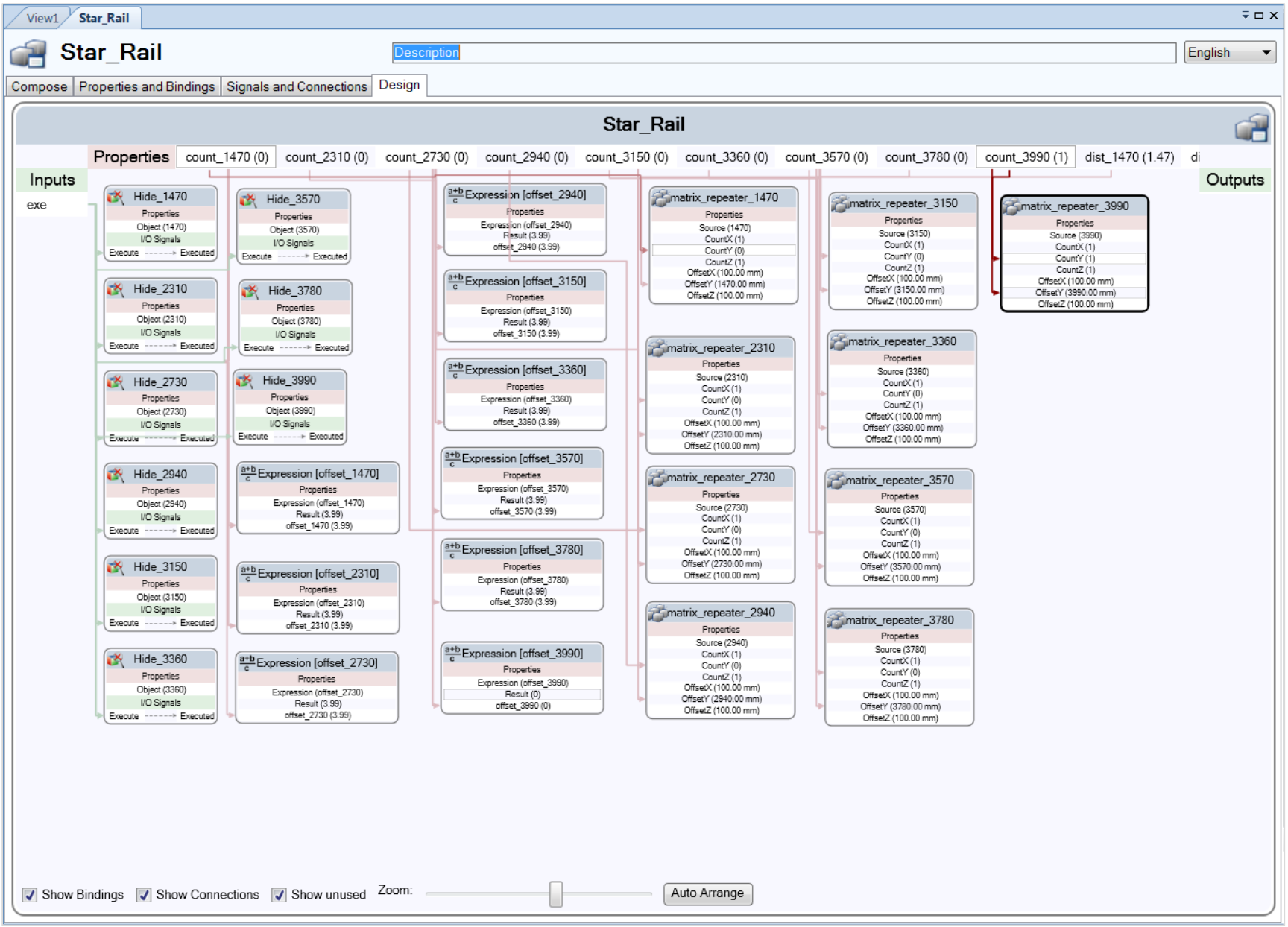
```
    <DynamicProperty name="dist_12ft_4ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_12ft_3ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_12ft_2ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
    <DynamicProperty name="dist_12ft_1ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>

    <DynamicProperty name="offset_12ft" valueType="System.Double" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="Length"/>
    </DynamicProperty>
</Properties>
<Bindings>
</Bindings>
<GraphicComponents>
</GraphicComponents>
<Assets>
                <Asset source="Fence_Calculator.dll"/>
</Assets>
</SmartComponent>
  </lc:Library>
</lc:LibraryCompiler>
```

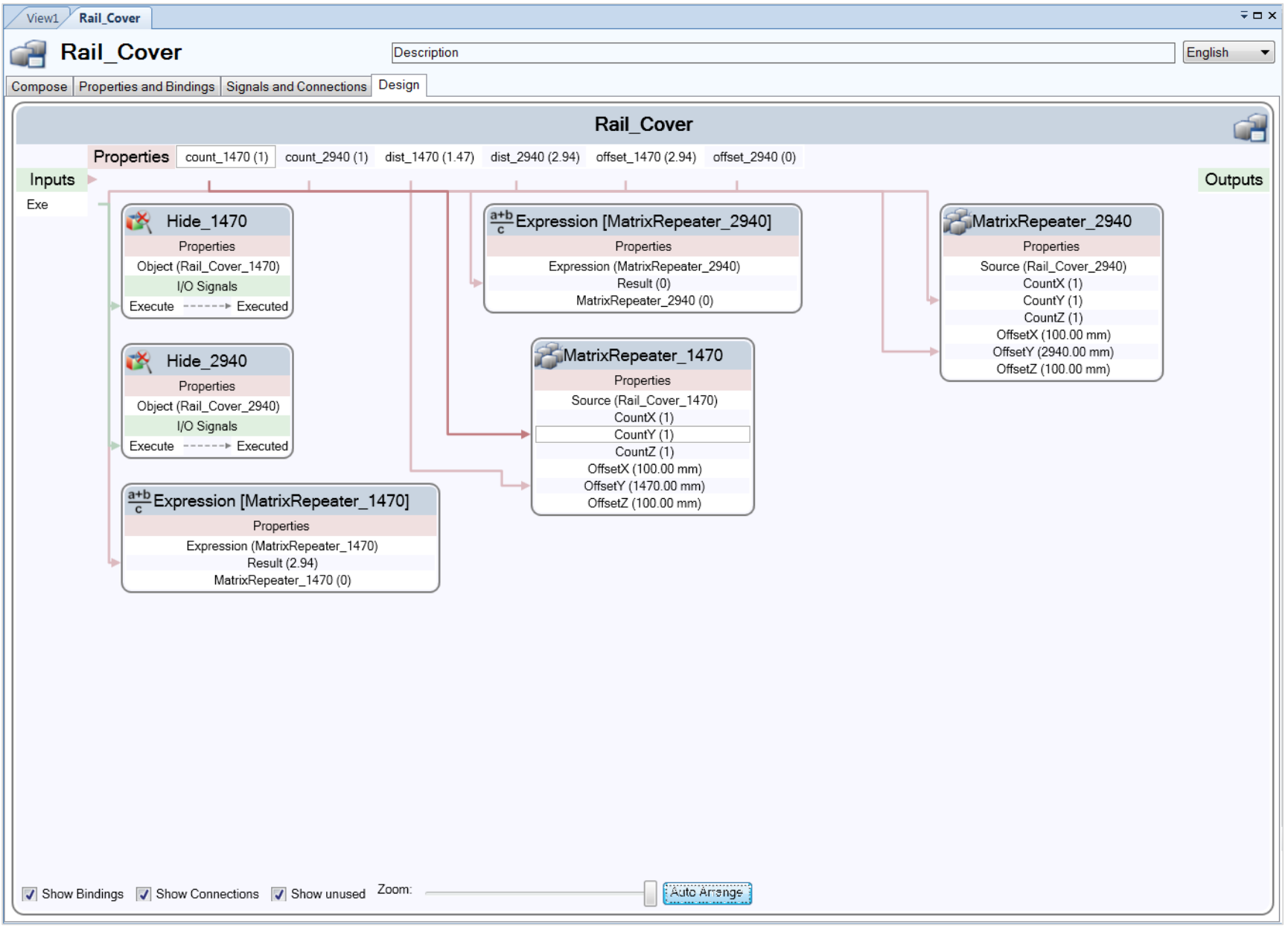
Track_Builder

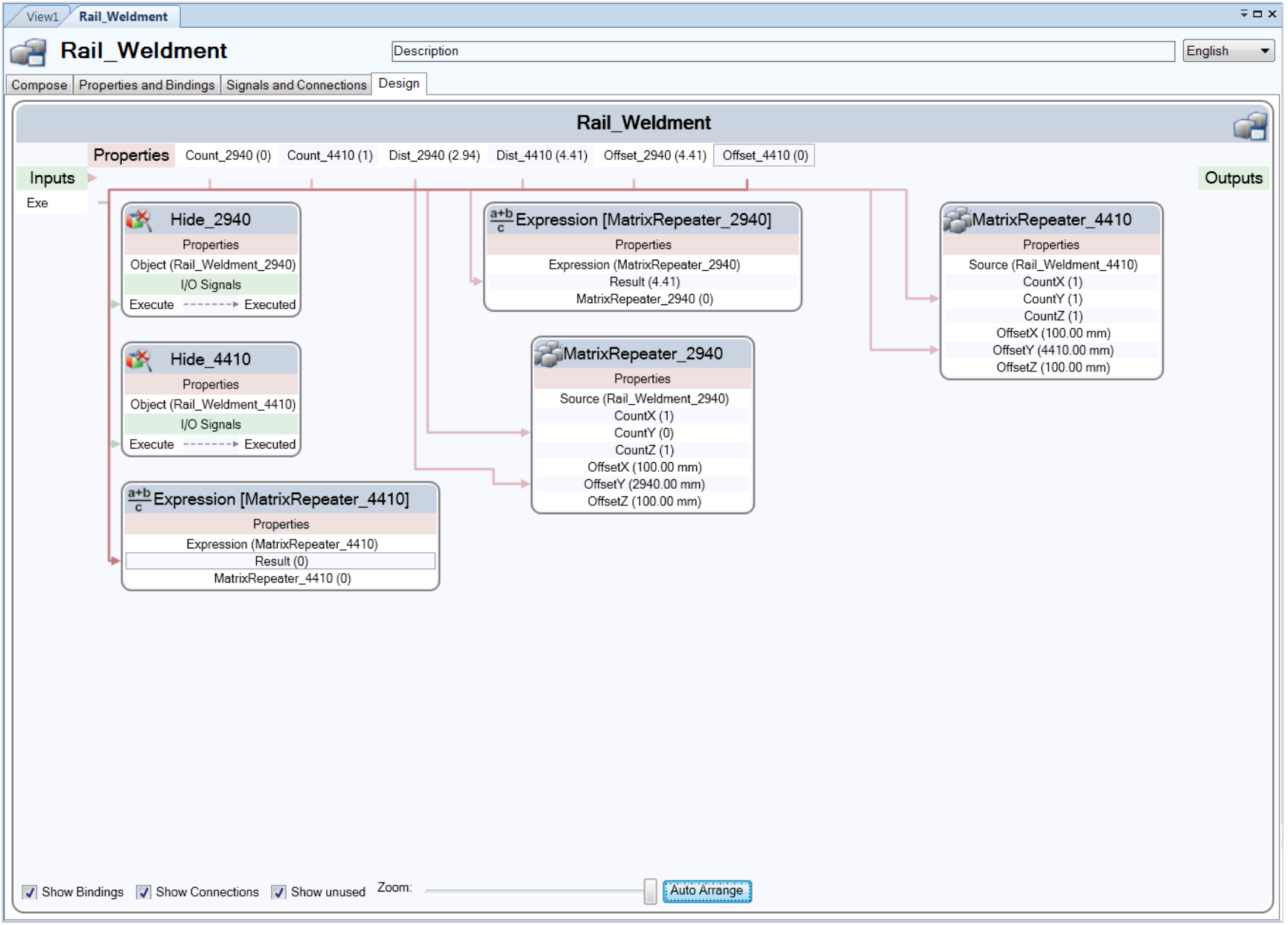
RobotStudio




View1
Rack
Rack
Description
English
Compose
Properties and Bindings
Signals and Connections
Design
Rack
Properties
count_1470 (0)
count_2940 (1)
dist_1470 (1.47)
dist_2940 (2.94)
offset_1470 (2.94)
offset_2940 (0)
Inputs
exe
Outputs
Hide_1470
Properties
Object (Rack_1470)
I/O Signals
Execute
Executed
Hide_2940
Properties
Object (Rack_2940)
I/O Signals
Execute
Executed
Expression [MatrixRepeater_1470]
Properties
Expression (MatrixRepeater_1470)
Result (2.94)
MatrixRepeater_1470 (0)
Expression [MatrixRepeater_2940]
Properties
Expression (MatrixRepeater_2940)
Result (0)
MatrixRepeater_2940 (0)
MatrixRepeater_1470
Properties
Source (Rack_1470)
CountX (1)
CountY (0)
CountZ (1)
OffsetX (100.00 mm)
OffsetY (1470.00 mm)
OffsetZ (100.00 mm)
MatrixRepeater_2940
Properties
Source (Rack_2940)
CountX (1)
CountY (1)
CountZ (1)
OffsetX (100.00 mm)
OffsetY (2940.00 mm)
OffsetZ (100.00 mm)
Show Bindings
Show Connections
Show unused
Zoom:
Auto Arrange


View1
Star_Rail
Star_Rail
Description
English
Compose
Properties and Bindings
Signals and Connections
Design
Star_Rail
Properties
count_1470 (0)
count_2310 (0)
count_2730 (0)
count_2940 (0)
count_3150 (0)
count_3360 (0)
count_3570 (0)
count_3780 (0)
count_3990 (1)
dist_1470 (1.47)
di
Inputs
exe
Outputs
Hide_1470
Hide_2310
Hide_2730
Hide_2940
Hide_3150
Hide_3360
Hide_3570
Hide_3780
Hide_3990
Expression [offset_1470]
Expression [offset_2310]
Expression [offset_2730]
Expression [offset_2940]
Expression [offset_3150]
Expression [offset_3360]
Expression [offset_3570]
Expression [offset_3780]
Expression [offset_3990]
matrix_repeater_1470
matrix_repeater_2310
matrix_repeater_2730
matrix_repeater_2940
matrix_repeater_3150
matrix_repeater_3360
matrix_repeater_3570
matrix_repeater_3780
matrix_repeater_3990
Show Bindings
Show Connections
Show unused
Zoom:
Auto Arrange


View1
Rail_Cover
Rail_Cover
Description
English
Compose
Properties and Bindings
Signals and Connections
Design
Rail_Cover
Properties
count_1470 (1)
count_2940 (1)
dist_1470 (1.47)
dist_2940 (2.94)
offset_1470 (2.94)
offset_2940 (0)
Inputs
Exe
Outputs
Hide_1470
Properties
Object (Rail_Cover_1470)
I/O Signals
Execute
Executed
Hide_2940
Properties
Object (Rail_Cover_2940)
I/O Signals
Execute
Executed
Expression [MatrixRepeater_2940]
Properties
Expression (MatrixRepeater_2940)
Result (0)
MatrixRepeater_2940 (0)
MatrixRepeater_1470
Properties
Source (Rail_Cover_1470)
CountX (1)
CountY (1)
CountZ (1)
OffsetX (100.00 mm)
OffsetY (1470.00 mm)
OffsetZ (100.00 mm)
MatrixRepeater_2940
Properties
Source (Rail_Cover_2940)
CountX (1)
CountY (1)
CountZ (1)
OffsetX (100.00 mm)
OffsetY (2940.00 mm)
OffsetZ (100.00 mm)
Expression [MatrixRepeater_1470]
Properties
Expression (MatrixRepeater_1470)
Result (2.94)
MatrixRepeater_1470 (0)
Show Bindings
Show Connections
Show unused
Zoom:
Auto Arrange


View1
Rail_Weldment
Rail_Weldment
Description
English
Compose
Properties and Bindings
Signals and Connections
Design
Rail_Weldment
Properties
Count_2940 (0)
Count_4410 (1)
Dist_2940 (2.94)
Dist_4410 (4.41)
Offset_2940 (4.41)
Offset_4410 (0)
Inputs
Exe
Outputs
Hide_2940
Properties
Object (Rail_Weldment_2940)
I/O Signals
Execute
Executed
Hide_4410
Properties
Object (Rail_Weldment_4410)
I/O Signals
Execute
Executed
Expression [MatrixRepeater_4410]
Properties
Expression (MatrixRepeater_4410)
Result (0)
MatrixRepeater_4410 (0)
Expression [MatrixRepeater_2940]
Properties
Expression (MatrixRepeater_2940)
Result (4.41)
MatrixRepeater_2940 (0)
MatrixRepeater_2940
Properties
Source (Rail_Weldment_2940)
CountX (1)
CountY (0)
CountZ (1)
OffsetX (100.00 mm)
OffsetY (2940.00 mm)
OffsetZ (100.00 mm)
MatrixRepeater_4410
Properties
Source (Rail_Weldment_4410)
CountX (1)
CountY (1)
CountZ (1)
OffsetX (100.00 mm)
OffsetY (4410.00 mm)
OffsetZ (100.00 mm)
Show Bindings
Show Connections
Show unused
Zoom:
Auto Arrange

CodeBehind

```
using System;
using System.Collections.Generic;
using System.Text;

using ABB.Robotics.Math;
using ABB.Robotics.RobotStudio;
using ABB.Robotics.RobotStudio.Stations;

namespace Track_Builder
{
   public class CodeBehind : SmartComponentCodeBehind
   {
      public void clear_mech()
      {
         GraphicComponent temp;
         Station station = Project.ActiveProject as Station;
         station.GraphicComponents.TryGetGraphicComponent("Carriage", out temp);
         if (temp != null)
         {
            station.GraphicComponents.Remove(temp);
         }

         //Clear Previous Accessories
         station.GraphicComponents.TryGetGraphicComponent("TorchClean_Bullseye", out temp);
         if (temp != null)
         {
            station.GraphicComponents.Remove(temp);
         }
         station.GraphicComponents.TryGetGraphicComponent("Robot_Cable_Management", out temp);
         if (temp != null)
         {
            station.GraphicComponents.Remove(temp);
         }
         station.GraphicComponents.TryGetGraphicComponent("Wire_Barrel", out temp);
         if (temp != null)
         {
            station.GraphicComponents.Remove(temp);
         }
         station.GraphicComponents.TryGetGraphicComponent("Water_Cooler", out temp);
         if (temp != null)
         {
            station.GraphicComponents.Remove(temp);
         }

         //Delete Previous Gantry
         station.GraphicComponents.TryGetGraphicComponent("Gantry_2500H", out temp);
         if (temp != null)
```

```
{
   station.GraphicComponents.Remove(temp);
}
station.GraphicComponents.TryGetGraphicComponent("Gantry_3500H", out temp);
if (temp != null)
{
   station.GraphicComponents.Remove(temp);
}
station.GraphicComponents.TryGetGraphicComponent("Gantry_4000H", out temp);
if (temp != null)
{
   station.GraphicComponents.Remove(temp);
}
station.GraphicComponents.TryGetGraphicComponent("Inv_Gantry_2500H", out temp);
if (temp != null)
{
   station.GraphicComponents.Remove(temp);
}
station.GraphicComponents.TryGetGraphicComponent("Inv_Gantry_3500H", out temp);
if (temp != null)
{
   station.GraphicComponents.Remove(temp);
}
station.GraphicComponents.TryGetGraphicComponent("Inv_Gantry_4000H", out temp);
if (temp != null)
{
   station.GraphicComponents.Remove(temp);
}

//Delete Previous Boom
station.GraphicComponents.TryGetGraphicComponent("Boom_1500", out temp);
if (temp != null)
{
   station.GraphicComponents.Remove(temp);
}
station.GraphicComponents.TryGetGraphicComponent("Boom_2000", out temp);
if (temp != null)
{
   station.GraphicComponents.Remove(temp);
}
station.GraphicComponents.TryGetGraphicComponent("Boom_2500", out temp);
if (temp != null)
{
   station.GraphicComponents.Remove(temp);
}
station.GraphicComponents.TryGetGraphicComponent("Boom_3000", out temp);
if (temp != null)
{
```

```
            station.GraphicComponents.Remove(temp);
        }
    }

    public override void OnPropertyValueChanged(SmartComponent component, DynamicProperty
changedProperty, Object oldValue)
    {
        bool name = !(changedProperty.Name == "Track_Length_mm");
        bool name2 = !(changedProperty.Name == "Gantry");
        bool name3 = !(changedProperty.Name == "Boom");
        bool name4 = !(changedProperty.Name == "RTT");
        bool name5 = !(changedProperty.Name == "Water_Cooler");
        bool name6 = !(changedProperty.Name == "Wire_Barrel");
        bool name7 = !(changedProperty.Name == "Torch_Cleaner_Bullseye");

        //Order of segments
        int[,] obj_data = new int[15, 20] {{ 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2 }, { 1, 0, 1, 2, 1,
2, 3, 2, 3, 4, 3, 4, 5, 4, 5, 6, 5, 6, 7, 6 }, { 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0 }, { 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1 }, { 0, 0, 1, 1, 2, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 1,
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 1, 1, 1, 1, 1, 2, 2, 3, 3, 3, 3, 4, 4, 5, 5, 5, 6, 6, 6, 6 }, { 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1, 0, 1, 0, 1, 0, 1, 0, 1, 0 }, { 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10, 11 }, { 0, 1, 0, 1, 0, 1, 2, 1, 2,
3, 2, 3, 4, 3, 4, 3, 4, 5, 6, 5 }, { 1, 1, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 7, 7, 7, 7, 8 }};

        //Check if Variables have changed
        if (!name | !name2 | !name3 | !name4 | !name5 | !name6 | !name7)
        {
            int Orig_Track_Length = (int)component.Properties["Track_Length_mm"].Value;
            string Orig_Gantry = (string)component.Properties["Gantry"].Value;
            string Orig_Boom = (string)component.Properties["Boom"].Value;
            string Orig_RTT = (string)component.Properties["RTT"].Value;

            string Orig_Water_Cooler = (string)component.Properties["Water_Cooler"].Value;
            string Orig_Wire_Barrel = (string)component.Properties["Wire_Barrel"].Value;
            string Orig_Torch_Cleaner_Bullseye =
(string)component.Properties["Torch_Cleaner_Bullseye"].Value;
            int n;

            if (Orig_RTT == "Robot")
            {
                //Remove previous mechanism if avalible
                clear_mech();

                //Load new mechanism and modify values
                Station station = Project.ActiveProject as Station;
                GraphicComponentLibrary mylib;
```

```
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Carriage\\Carriage.rslib", true);
            Mechanism Gantry = (Mechanism)mylib.RootComponent.CopyInstance();
            Gantry.Name = "Carriage";
            Gantry.SetJointLimits(0, 0, ((Orig_Track_Length — 1667.05) / 1000));
            station.GraphicComponents.Add(Gantry);
            Gantry.DisconnectFromLibrary();
          }

          if (Orig_Gantry == "Gantry_2500H")
          {
            //Remove revious mechanism if avalible
            clear_mech();

            //Load mechanism tower
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Gantry_2500H.rslib", true);
            Mechanism Gantry = (Mechanism)mylib.RootComponent.CopyInstance();
            Gantry.Name = "Gantry_2500H";
            Gantry.SetJointLimits(0, 0, ((Orig_Track_Length — 1667.05) / 1000));
            station.GraphicComponents.Add(Gantry);
            Gantry.DisconnectFromLibrary();

            //Load Water Cooler
            if (Orig_Water_Cooler == "Lincoln")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Lincoln_Water_Cooler.rslib", true);
              Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
              Water_Cooler.Name = "Water_Cooler";
              station.GraphicComponents.Add(Water_Cooler);

              //Attach boom to mechanism
              GraphicComponent L3;
              Gantry.GetParentLink(2, out L3);
              Part temp = (Part)L3;
              temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

              Water_Cooler.DisconnectFromLibrary();
            }

            if (Orig_Water_Cooler == "Miller")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Miller_Water_Cooler.rslib", true);
              Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
```

```
            Water_Cooler.Name = "Water_Cooler";
            station.GraphicComponents.Add(Water_Cooler);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

            Water_Cooler.DisconnectFromLibrary();
        }

        //Load Wire Barrel
        if (Orig_Wire_Barrel == "Single_Wire_Barrel_Low")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Single_Wire_Barrel_Low_Mount.rslib", true);
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
        }
        if (Orig_Wire_Barrel == "Tandum_Wire_Barrel_Low")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Tandum_Wire_Barrel_Low_Mount.rslib", true);
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
        }
        if (Orig_Wire_Barrel == "Wire_Barrel_High")
        {
```

```
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Wire_Barrel_High_Short_Mount.rslib", true);
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
          }

          //Load Torch Cleaner and Bullseye
          if (Orig_Torch_Cleaner_Bullseye == "yes")
          {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\Robot_Cable_Management.rslib", true);
            Part Robot_Cable_Management = (Part)mylib.RootComponent.CopyInstance();
            Robot_Cable_Management.Name = "Robot_Cable_Management";
            station.GraphicComponents.Add(Robot_Cable_Management);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Robot_Cable_Management, false, new Matrix4(new Vector3(0, 0, 0)));

            Robot_Cable_Management.DisconnectFromLibrary();

            if(Orig_Boom == "Boom_1500")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_1500.rslib", true);
              Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
              TorchClean_Bullseye.Name = "TorchClean_Bullseye";
              station.GraphicComponents.Add(TorchClean_Bullseye);

              //Attach boom to mechanism
              temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

              TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_2000")
```

```
            {
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2000.rslib", true);
                Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
                TorchClean_Bullseye.Name = "TorchClean_Bullseye";
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_2500")
            {
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2500.rslib", true);
                Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
                TorchClean_Bullseye.Name = "TorchClean_Bullseye";
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_3000")
            {
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_3000.rslib", true);
                Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
                TorchClean_Bullseye.Name = "TorchClean_Bullseye";
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
            }
        }

        //Load Boom
        if (Orig_Boom == "Boom_1500")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_1500.rslib", true);
```

```
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_1500";
            station.GraphicComponents.Add(Boom);
            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .017921;
            Boom.Transform.Z = .971197;
            Boom.Transform.RX = 0;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);


            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_2000")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2000.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_2000";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = .971197;
            Boom.Transform.RX = 0;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_2500")
```

```
            {
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2500.rslib", true);
                Part Boom = (Part)mylib.RootComponent.CopyInstance();
                Boom.Name = "Boom_2500";
                Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
                station.GraphicComponents.Add(Boom);

                //Place boom in correct position
                Boom.Transform.X = 0;
                Boom.Transform.Y = .000521;
                Boom.Transform.Z = .971197;
                Boom.Transform.RX = 0;
                Boom.Transform.RY = 0;
                Boom.Transform.RZ = (3.14 / 2);

                //Attach boom to mechanism
                GraphicComponent L4;
                Gantry.GetParentLink(3, out L4);
                Part temp = (Part)L4;
                temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

                Boom.DisconnectFromLibrary();
            }

            if (Orig_Boom == "Boom_3000")
            {
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_3000.rslib", true);
                Part Boom = (Part)mylib.RootComponent.CopyInstance();
                Boom.Name = "Boom_3000";
                Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
                station.GraphicComponents.Add(Boom);

                //Place boom in correct position
                Boom.Transform.X = 0;
                Boom.Transform.Y = .000521;
                Boom.Transform.Z = .971197;
                Boom.Transform.RX = 0;
                Boom.Transform.RY = 0;
                Boom.Transform.RZ = (3.14 / 2);

                //Attach boom to mechanism
                GraphicComponent L4;
                Gantry.GetParentLink(3, out L4);
                Part temp = (Part)L4;
                temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));
```

```
            Boom.DisconnectFromLibrary();
        }
    }

    if (Orig_Gantry == "Gantry_3500H")
    {
        //Remove previous mechanism if avalible
        clear_mech();

        //Load mechanism tower
        Station station = Project.ActiveProject as Station;
        GraphicComponentLibrary mylib;
        mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Gantry_3500H.rslib", true);
        Mechanism Gantry = (Mechanism)mylib.RootComponent.CopyInstance();
        Gantry.Name = "Gantry_3500H";
        Gantry.SetJointLimits(0, 0, ((Orig_Track_Length — 1667.05) / 1000));
        station.GraphicComponents.Add(Gantry);
        Gantry.DisconnectFromLibrary();

        //Load Water Cooler
        if (Orig_Water_Cooler == "Lincoln")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Lincoln_Water_Cooler.rslib", true);
            Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
            Water_Cooler.Name = "Water_Cooler";
            station.GraphicComponents.Add(Water_Cooler);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

            Water_Cooler.DisconnectFromLibrary();
        }

        if (Orig_Water_Cooler == "Miller")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Miller_Water_Cooler.rslib", true);
            Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
            Water_Cooler.Name = "Water_Cooler";
            station.GraphicComponents.Add(Water_Cooler);

            //Attach boom to mechanism
            GraphicComponent L3;
```

```
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

            Water_Cooler.DisconnectFromLibrary();
        }

        //Load Wire Barrel
        if (Orig_Wire_Barrel == "Single_Wire_Barrel_Low")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Single_Wire_Barrel_Low_Mount.rslib", true);
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
        }
        if (Orig_Wire_Barrel == "Tandum_Wire_Barrel_Low")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Tandum_Wire_Barrel_Low_Mount.rslib", true);
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
        }
        if (Orig_Wire_Barrel == "Wire_Barrel_High")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Wire_Barrel_High_Short_Mount.rslib", true);
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);
```

```
            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
          }

          //Load Torch Cleaner and Bullseye
          if (Orig_Torch_Cleaner_Bullseye == "yes")
          {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\Robot_Cable_Management.rslib", true);
            Part Robot_Cable_Management = (Part)mylib.RootComponent.CopyInstance();
            Robot_Cable_Management.Name = "Robot_Cable_Management";
            station.GraphicComponents.Add(Robot_Cable_Management);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Robot_Cable_Management, false, new Matrix4(new Vector3(0, 0, 0)));

            Robot_Cable_Management.DisconnectFromLibrary();

            if (Orig_Boom == "Boom_1500")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_1500.rslib", true);
              Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
              TorchClean_Bullseye.Name = "TorchClean_Bullseye";
              station.GraphicComponents.Add(TorchClean_Bullseye);

              //Attach boom to mechanism
              temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

              TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_2000")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2000.rslib", true);
              Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
              TorchClean_Bullseye.Name = "TorchClean_Bullseye";
```

```
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_2500")
            {
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2500.rslib", true);
                Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
                TorchClean_Bullseye.Name = "TorchClean_Bullseye";
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_3000")
            {
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_3000.rslib", true);
                Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
                TorchClean_Bullseye.Name = "TorchClean_Bullseye";
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
            }
        }

        //Load Boom
        if (Orig_Boom == "Boom_1500")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_1500.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_1500";
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
```

```
            Boom.Transform.Y = .017921;
            Boom.Transform.Z = .971197;
            Boom.Transform.RX = 0;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_2000")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2000.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_2000";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = .971197;
            Boom.Transform.RX = 0;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_2500")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2500.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_2500";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
```

```
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = .971197;
            Boom.Transform.RX = 0;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
          }

          if (Orig_Boom == "Boom_3000")
          {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_3000.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_3000";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = .971197;
            Boom.Transform.RX = 0;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
          }
        }

        if (Orig_Gantry == "Gantry_4000H")
        {
```

```
            //Remove previous Mechanism if avalible
            clear_mech();

            //Load mechanism tower
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Gantry_4000H.rslib", true);
            Mechanism Gantry = (Mechanism)mylib.RootComponent.CopyInstance();
            Gantry.Name = "Gantry_4000H";
            Gantry.SetJointLimits(0, 0, ((Orig_Track_Length — 1667.05) / 1000));
            station.GraphicComponents.Add(Gantry);
            Gantry.DisconnectFromLibrary();

            //Load Water Cooler
            if (Orig_Water_Cooler == "Lincoln")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Lincoln_Water_Cooler.rslib", true);
               Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
               Water_Cooler.Name = "Water_Cooler";
               station.GraphicComponents.Add(Water_Cooler);

               //Attach boom to mechanism
               GraphicComponent L3;
               Gantry.GetParentLink(2, out L3);
               Part temp = (Part)L3;
               temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

               Water_Cooler.DisconnectFromLibrary();
            }

            if (Orig_Water_Cooler == "Miller")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Miller_Water_Cooler.rslib", true);
               Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
               Water_Cooler.Name = "Water_Cooler";
               station.GraphicComponents.Add(Water_Cooler);

               //Attach boom to mechanism
               GraphicComponent L3;
               Gantry.GetParentLink(2, out L3);
               Part temp = (Part)L3;
               temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

               Water_Cooler.DisconnectFromLibrary();
            }
```

```
            //Load Wire Barrel
            if (Orig_Wire_Barrel == "Single_Wire_Barrel_Low")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Single_Wire_Barrel_Low_Mount.rslib", true);
               Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
               Wire_Barrel.Name = "Wire_Barrel";
               station.GraphicComponents.Add(Wire_Barrel);

               //Attach boom to mechanism
               GraphicComponent L3;
               Gantry.GetParentLink(2, out L3);
               Part temp = (Part)L3;
               temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

               Wire_Barrel.DisconnectFromLibrary();
            }
            if (Orig_Wire_Barrel == "Tandum_Wire_Barrel_Low")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Tandum_Wire_Barrel_Low_Mount.rslib", true);
               Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
               Wire_Barrel.Name = "Wire_Barrel";
               station.GraphicComponents.Add(Wire_Barrel);

               //Attach boom to mechanism
               GraphicComponent L3;
               Gantry.GetParentLink(2, out L3);
               Part temp = (Part)L3;
               temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

               Wire_Barrel.DisconnectFromLibrary();
            }
            if (Orig_Wire_Barrel == "Wire_Barrel_High")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Wire_Barrel_High_Short_Mount.rslib", true);
               Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
               Wire_Barrel.Name = "Wire_Barrel";
               station.GraphicComponents.Add(Wire_Barrel);

               //Attach boom to mechanism
               GraphicComponent L3;
               Gantry.GetParentLink(2, out L3);
               Part temp = (Part)L3;
               temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));
```

```
            Wire_Barrel.DisconnectFromLibrary();
        }

        //Load Torch Cleaner and Bullseye
        if (Orig_Torch_Cleaner_Bullseye == "yes")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\Robot_Cable_Management.rslib", true);
            Part Robot_Cable_Management = (Part)mylib.RootComponent.CopyInstance();
            Robot_Cable_Management.Name = "Robot_Cable_Management";
            station.GraphicComponents.Add(Robot_Cable_Management);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Robot_Cable_Management, false, new Matrix4(new Vector3(0, 0, 0)));

            Robot_Cable_Management.DisconnectFromLibrary();

            if (Orig_Boom == "Boom_1500")
            {
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_1500.rslib", true);
                Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
                TorchClean_Bullseye.Name = "TorchClean_Bullseye";
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_2000")
            {
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2000.rslib", true);
                Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
                TorchClean_Bullseye.Name = "TorchClean_Bullseye";
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
```

```
            }
            if (Orig_Boom == "Boom_2500")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2500.rslib", true);
              Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
              TorchClean_Bullseye.Name = "TorchClean_Bullseye";
              station.GraphicComponents.Add(TorchClean_Bullseye);

              //Attach boom to mechanism
              temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

              TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_3000")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_3000.rslib", true);
              Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
              TorchClean_Bullseye.Name = "TorchClean_Bullseye";
              station.GraphicComponents.Add(TorchClean_Bullseye);

              //Attach boom to mechanism
              temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

              TorchClean_Bullseye.DisconnectFromLibrary();
            }
          }

          //Load Boom
          if (Orig_Boom == "Boom_1500")
          {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_1500.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_1500";
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .017921;
            Boom.Transform.Z = .971197;
            Boom.Transform.RX = 0;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);
```

```
            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_2000")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2000.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_2000";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = .971197;
            Boom.Transform.RX = 0;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_2500")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2500.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_2500";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = .971197;
```

```
            Boom.Transform.RX = 0;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_3000")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_3000.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_3000";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = .971197;
            Boom.Transform.RX = 0;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }
    }

    if (Orig_Gantry == "Inv_Gantry_2500H")
    {
        //Remove previous Mechanism if avalible
        clear_mech();

        //Load mechanism tower
        Station station = Project.ActiveProject as Station;
        GraphicComponentLibrary mylib;
```

```
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Inv_Gantry_2500H.rslib", true);
            Mechanism Gantry = (Mechanism)mylib.RootComponent.CopyInstance();
            Gantry.Name = "Inv_Gantry_2500H";
            Gantry.SetJointLimits(0, 0, ((Orig_Track_Length — 1667.05) / 1000));
            station.GraphicComponents.Add(Gantry);
            Gantry.DisconnectFromLibrary();

            //Load Water Cooler
            if (Orig_Water_Cooler == "Lincoln")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Lincoln_Water_Cooler.rslib", true);
               Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
               Water_Cooler.Name = "Water_Cooler";
               station.GraphicComponents.Add(Water_Cooler);

               //Attach boom to mechanism
               GraphicComponent L3;
               Gantry.GetParentLink(2, out L3);
               Part temp = (Part)L3;
               temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

               Water_Cooler.DisconnectFromLibrary();
            }

            if (Orig_Water_Cooler == "Miller")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Miller_Water_Cooler.rslib", true);
               Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
               Water_Cooler.Name = "Water_Cooler";
               station.GraphicComponents.Add(Water_Cooler);

               //Attach boom to mechanism
               GraphicComponent L3;
               Gantry.GetParentLink(2, out L3);
               Part temp = (Part)L3;
               temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

               Water_Cooler.DisconnectFromLibrary();
            }

            //Load Wire Barrel
            if (Orig_Wire_Barrel == "Single_Wire_Barrel_Low")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Single_Wire_Barrel_Low_Mount.rslib", true);
```

```
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
        }
        if (Orig_Wire_Barrel == "Tandum_Wire_Barrel_Low")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Tandum_Wire_Barrel_Low_Mount.rslib", true);
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
        }
        if (Orig_Wire_Barrel == "Wire_Barrel_High")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Wire_Barrel_High_Short_Mount.rslib", true);
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
        }

        //Load Torch Cleaner and Bullseye
        if (Orig_Torch_Cleaner_Bullseye == "yes")
        {
```

```
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\Robot_Cable_Management.rslib", true);
            Part Robot_Cable_Management = (Part)mylib.RootComponent.CopyInstance();
            Robot_Cable_Management.Name = "Robot_Cable_Management";
            station.GraphicComponents.Add(Robot_Cable_Management);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Robot_Cable_Management, false, new Matrix4(new Vector3(0, 0, 0)));

            Robot_Cable_Management.DisconnectFromLibrary();

            if (Orig_Boom == "Boom_1500")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_1500.rslib", true);
              Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
              TorchClean_Bullseye.Name = "TorchClean_Bullseye";
              station.GraphicComponents.Add(TorchClean_Bullseye);

              //Attach boom to mechanism
              temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

              TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_2000")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2000.rslib", true);
              Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
              TorchClean_Bullseye.Name = "TorchClean_Bullseye";
              station.GraphicComponents.Add(TorchClean_Bullseye);

              //Attach boom to mechanism
              temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

              TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_2500")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2500.rslib", true);
```

```
                Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
                TorchClean_Bullseye.Name = "TorchClean_Bullseye";
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_3000")
            {
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_3000.rslib", true);
                Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
                TorchClean_Bullseye.Name = "TorchClean_Bullseye";
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
            }
        }

        //Load Boom
        if (Orig_Boom == "Boom_1500")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_1500.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_1500";
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .017921;
            Boom.Transform.Z = 1.168683;
            Boom.Transform.RX = 3.14;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14/2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));
```

```
            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_2000")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2000.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_2000";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = 1.168683;
            Boom.Transform.RX = 3.14;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_2500")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2500.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_2500";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = 1.168683;
            Boom.Transform.RX = 3.14;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
```

```
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
          }

          if (Orig_Boom == "Boom_3000")
          {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_3000.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_3000";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = 1.168683;
            Boom.Transform.RX = 3.14;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
          }
        }

        if (Orig_Gantry == "Inv_Gantry_3500H")
        {
          //Remove previous Mechanism if avalible
          clear_mech();

          //Load mechanism tower
          Station station = Project.ActiveProject as Station;
          GraphicComponentLibrary mylib;
          mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Inv_Gantry_3500H.rslib", true);
          Mechanism Gantry = (Mechanism)mylib.RootComponent.CopyInstance();
          Gantry.Name = "Inv_Gantry_3500H";
          Gantry.SetJointLimits(0, 0, ((Orig_Track_Length — 1667.05) / 1000));
          station.GraphicComponents.Add(Gantry);
```

```
            Gantry.DisconnectFromLibrary();

            //Load Water Cooler
            if (Orig_Water_Cooler == "Lincoln")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Lincoln_Water_Cooler.rslib", true);
               Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
               Water_Cooler.Name = "Water_Cooler";
               station.GraphicComponents.Add(Water_Cooler);

               //Attach boom to mechanism
               GraphicComponent L3;
               Gantry.GetParentLink(2, out L3);
               Part temp = (Part)L3;
               temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

               Water_Cooler.DisconnectFromLibrary();
            }

            if (Orig_Water_Cooler == "Miller")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Miller_Water_Cooler.rslib", true);
               Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
               Water_Cooler.Name = "Water_Cooler";
               station.GraphicComponents.Add(Water_Cooler);

               //Attach boom to mechanism
               GraphicComponent L3;
               Gantry.GetParentLink(2, out L3);
               Part temp = (Part)L3;
               temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

               Water_Cooler.DisconnectFromLibrary();
            }

            //Load Wire Barrel
            if (Orig_Wire_Barrel == "Single_Wire_Barrel_Low")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Single_Wire_Barrel_Low_Mount.rslib", true);
               Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
               Wire_Barrel.Name = "Wire_Barrel";
               station.GraphicComponents.Add(Wire_Barrel);

               //Attach boom to mechanism
               GraphicComponent L3;
```

```
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
          }
          if (Orig_Wire_Barrel == "Tandum_Wire_Barrel_Low")
          {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Tandum_Wire_Barrel_Low_Mount.rslib", true);
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
          }
          if (Orig_Wire_Barrel == "Wire_Barrel_High")
          {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Wire_Barrel_High_Short_Mount.rslib", true);
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
          }

          //Load Torch Cleaner and Bullseye
          if (Orig_Torch_Cleaner_Bullseye == "yes")
          {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\Robot_Cable_Management.rslib", true);
            Part Robot_Cable_Management = (Part)mylib.RootComponent.CopyInstance();
            Robot_Cable_Management.Name = "Robot_Cable_Management";
            station.GraphicComponents.Add(Robot_Cable_Management);
```

```
            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Robot_Cable_Management, false, new Matrix4(new Vector3(0, 0, 0)));

            Robot_Cable_Management.DisconnectFromLibrary();

            if (Orig_Boom == "Boom_1500")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_1500.rslib", true);
               Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
               TorchClean_Bullseye.Name = "TorchClean_Bullseye";
               station.GraphicComponents.Add(TorchClean_Bullseye);

               //Attach boom to mechanism
               temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

               TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_2000")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2000.rslib", true);
               Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
               TorchClean_Bullseye.Name = "TorchClean_Bullseye";
               station.GraphicComponents.Add(TorchClean_Bullseye);

               //Attach boom to mechanism
               temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

               TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_2500")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2500.rslib", true);
               Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
               TorchClean_Bullseye.Name = "TorchClean_Bullseye";
               station.GraphicComponents.Add(TorchClean_Bullseye);

               //Attach boom to mechanism
               temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));
```

```
                TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_3000")
            {
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_3000.rslib", true);
                Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
                TorchClean_Bullseye.Name = "TorchClean_Bullseye";
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
            }
        }

        //Load Boom
        if (Orig_Boom == "Boom_1500")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_1500.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_1500";
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .017921;
            Boom.Transform.Z = 1.308536;
            Boom.Transform.RX = 3.14;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_2000")
        {
```

```
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2000.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_2000";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = 1.308536;
            Boom.Transform.RX = 3.14;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
          }

          if (Orig_Boom == "Boom_2500")
          {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2500.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_2500";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = 1.308536;
            Boom.Transform.RX = 3.14;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
```

```
            }

            if (Orig_Boom == "Boom_3000")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_3000.rslib", true);
               Part Boom = (Part)mylib.RootComponent.CopyInstance();
               Boom.Name = "Boom_3000";
               Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
               station.GraphicComponents.Add(Boom);

               //Place boom in correct position
               Boom.Transform.X = 0;
               Boom.Transform.Y = .000521;
               Boom.Transform.Z = 1.308536;
               Boom.Transform.RX = 3.14;
               Boom.Transform.RY = 0;
               Boom.Transform.RZ = (3.14 / 2);

               //Attach boom to mechanism
               GraphicComponent L4;
               Gantry.GetParentLink(3, out L4);
               Part temp = (Part)L4;
               temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

               Boom.DisconnectFromLibrary();
            }
         }

         if (Orig_Gantry == "Inv_Gantry_4000H")
         {
            //Remove previous Mechanism if avalible
            clear_mech();

            //Load mechanism tower
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Inv_Gantry_4000H.rslib", true);
            Mechanism Gantry = (Mechanism)mylib.RootComponent.CopyInstance();
            Gantry.Name = "Inv_Gantry_4000H";
            Gantry.SetJointLimits(0, 0, ((Orig_Track_Length — 1667.05) / 1000));
            station.GraphicComponents.Add(Gantry);
            Gantry.DisconnectFromLibrary();

            //Load Water Cooler
            if (Orig_Water_Cooler == "Lincoln")
            {
```

```
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Lincoln_Water_Cooler.rslib", true);
            Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
            Water_Cooler.Name = "Water_Cooler";
            station.GraphicComponents.Add(Water_Cooler);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

            Water_Cooler.DisconnectFromLibrary();
          }

          if (Orig_Water_Cooler == "Miller")
          {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Water Cooler\\Miller_Water_Cooler.rslib", true);
            Part Water_Cooler = (Part)mylib.RootComponent.CopyInstance();
            Water_Cooler.Name = "Water_Cooler";
            station.GraphicComponents.Add(Water_Cooler);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Water_Cooler, false, new Matrix4(new Vector3(0, 0, 0)));

            Water_Cooler.DisconnectFromLibrary();
          }

          //Load Wire Barrel
          if (Orig_Wire_Barrel == "Single_Wire_Barrel_Low")
          {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Single_Wire_Barrel_Low_Mount.rslib", true);
            Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
            Wire_Barrel.Name = "Wire_Barrel";
            station.GraphicComponents.Add(Wire_Barrel);

            //Attach boom to mechanism
            GraphicComponent L3;
            Gantry.GetParentLink(2, out L3);
            Part temp = (Part)L3;
            temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

            Wire_Barrel.DisconnectFromLibrary();
```

```
            }
            if (Orig_Wire_Barrel == "Tandum_Wire_Barrel_Low")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Tandum_Wire_Barrel_Low_Mount.rslib", true);
               Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
               Wire_Barrel.Name = "Wire_Barrel";
               station.GraphicComponents.Add(Wire_Barrel);

               //Attach boom to mechanism
               GraphicComponent L3;
               Gantry.GetParentLink(2, out L3);
               Part temp = (Part)L3;
               temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

               Wire_Barrel.DisconnectFromLibrary();
            }
            if (Orig_Wire_Barrel == "Wire_Barrel_High")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Wire Barrel\\Wire_Barrel_High_Short_Mount.rslib", true);
               Part Wire_Barrel = (Part)mylib.RootComponent.CopyInstance();
               Wire_Barrel.Name = "Wire_Barrel";
               station.GraphicComponents.Add(Wire_Barrel);

               //Attach boom to mechanism
               GraphicComponent L3;
               Gantry.GetParentLink(2, out L3);
               Part temp = (Part)L3;
               temp.Attach(Wire_Barrel, false, new Matrix4(new Vector3(0, 0, 0)));

               Wire_Barrel.DisconnectFromLibrary();
            }

            //Load Torch Cleaner and Bullseye
            if (Orig_Torch_Cleaner_Bullseye == "yes")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\Robot_Cable_Management.rslib", true);
               Part Robot_Cable_Management = (Part)mylib.RootComponent.CopyInstance();
               Robot_Cable_Management.Name = "Robot_Cable_Management";
               station.GraphicComponents.Add(Robot_Cable_Management);

               //Attach boom to mechanism
               GraphicComponent L4;
               Gantry.GetParentLink(3, out L4);
               Part temp = (Part)L4;
```

```
            temp.Attach(Robot_Cable_Management, false, new Matrix4(new Vector3(0, 0, 0)));

            Robot_Cable_Management.DisconnectFromLibrary();

            if (Orig_Boom == "Boom_1500")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_1500.rslib", true);
               Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
               TorchClean_Bullseye.Name = "TorchClean_Bullseye";
               station.GraphicComponents.Add(TorchClean_Bullseye);

               //Attach boom to mechanism
               temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

               TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_2000")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2000.rslib", true);
               Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
               TorchClean_Bullseye.Name = "TorchClean_Bullseye";
               station.GraphicComponents.Add(TorchClean_Bullseye);

               //Attach boom to mechanism
               temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

               TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_2500")
            {
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_2500.rslib", true);
               Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
               TorchClean_Bullseye.Name = "TorchClean_Bullseye";
               station.GraphicComponents.Add(TorchClean_Bullseye);

               //Attach boom to mechanism
               temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

               TorchClean_Bullseye.DisconnectFromLibrary();
            }
            if (Orig_Boom == "Boom_3000")
            {
```

```
                mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Accessories\\Cable Management-Torch
Clean\\TorchClean_Bullseye_3000.rslib", true);
                Part TorchClean_Bullseye = (Part)mylib.RootComponent.CopyInstance();
                TorchClean_Bullseye.Name = "TorchClean_Bullseye";
                station.GraphicComponents.Add(TorchClean_Bullseye);

                //Attach boom to mechanism
                temp.Attach(TorchClean_Bullseye, false, new Matrix4(new Vector3(0, 0, 0)));

                TorchClean_Bullseye.DisconnectFromLibrary();
              }
            }

            //Load Boom
            if (Orig_Boom == "Boom_1500")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_1500.rslib", true);
              Part Boom = (Part)mylib.RootComponent.CopyInstance();
              Boom.Name = "Boom_1500";
              station.GraphicComponents.Add(Boom);

              //Place boom in correct position
              Boom.Transform.X = 0;
              Boom.Transform.Y = .017921;
              Boom.Transform.Z = 1.308536;
              Boom.Transform.RX = 3.14;
              Boom.Transform.RY = 0;
              Boom.Transform.RZ = (3.14 / 2);

              //Attach boom to mechanism
              GraphicComponent L4;
              Gantry.GetParentLink(3, out L4);
              Part temp = (Part)L4;
              temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

              Boom.DisconnectFromLibrary();
            }

            if (Orig_Boom == "Boom_2000")
            {
              mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2000.rslib", true);
              Part Boom = (Part)mylib.RootComponent.CopyInstance();
              Boom.Name = "Boom_2000";
              Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
              station.GraphicComponents.Add(Boom);
```

```
            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = 1.308536;
            Boom.Transform.RX = 3.14;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_2500")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_2500.rslib", true);
            Part Boom = (Part)mylib.RootComponent.CopyInstance();
            Boom.Name = "Boom_2500";
            Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
            station.GraphicComponents.Add(Boom);

            //Place boom in correct position
            Boom.Transform.X = 0;
            Boom.Transform.Y = .000521;
            Boom.Transform.Z = 1.308536;
            Boom.Transform.RX = 3.14;
            Boom.Transform.RY = 0;
            Boom.Transform.RZ = (3.14 / 2);

            //Attach boom to mechanism
            GraphicComponent L4;
            Gantry.GetParentLink(3, out L4);
            Part temp = (Part)L4;
            temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

            Boom.DisconnectFromLibrary();
        }

        if (Orig_Boom == "Boom_3000")
        {
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Gantry\\Boom\\Boom_3000.rslib", true);
```

```
        Part Boom = (Part)mylib.RootComponent.CopyInstance();
        Boom.Name = "Boom_3000";
        Boom.Transform.GlobalMatrix.Translate(708.271, 666.345, 946.033);
        station.GraphicComponents.Add(Boom);

        //Place boom in correct position
        Boom.Transform.X = 0;
        Boom.Transform.Y = .000521;
        Boom.Transform.Z = 1.308536;
        Boom.Transform.RX = 3.14;
        Boom.Transform.RY = 0;
        Boom.Transform.RZ = (3.14 / 2);

        //Attach boom to mechanism
        GraphicComponent L4;
        Gantry.GetParentLink(3, out L4);
        Part temp = (Part)L4;
        temp.Attach(Boom, false, new Matrix4(new Vector3(0, 0, 0)));

        Boom.DisconnectFromLibrary();
    }
}

// Calculate offsets and output variables to Robot Studio
if (Orig_Track_Length == 4410)
{
    n = 0;
    component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
    component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
    component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
    component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
    component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
    component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
    component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
    component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
    component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
    component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
    component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
    component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
    component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
    component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
    component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

    component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
    component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
    component.Properties["Star_Rail_1470_dist"].Value = (1.470);
    component.Properties["Star_Rail_2310_dist"].Value = (2.310);
    component.Properties["Star_Rail_2730_dist"].Value = (2.730);
```

```
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 5880)
        {
            n = 1;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);
```

```
            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 7350)
        {
            n = 2;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
```

```
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 8820)
        {
            n = 3;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
```

```
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = ((3.990 * obj_data[10, n]) + (2.940 *
obj_data[2,n]));
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 10290)
        {
            n = 4;
```

```
component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
component.Properties["Star_Rail_1470_dist"].Value = (1.470);
component.Properties["Star_Rail_2310_dist"].Value = (2.310);
component.Properties["Star_Rail_2730_dist"].Value = (2.730);
component.Properties["Star_Rail_2940_dist"].Value = (2.940);
component.Properties["Star_Rail_3150_dist"].Value = (3.150);
component.Properties["Star_Rail_3360_dist"].Value = (3.360);
component.Properties["Star_Rail_3570_dist"].Value = (3.570);
component.Properties["Star_Rail_3780_dist"].Value = (3.780);
component.Properties["Star_Rail_3990_dist"].Value = (3.990);
component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
component.Properties["Rack_1470_dist"].Value = (1.470);
component.Properties["Rack_2940_dist"].Value = (2.940);

component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
component.Properties["Star_Rail_3990_offset"].Value = (0);
component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
component.Properties["Rail_Cover_2940_offset"].Value = (0);
component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
component.Properties["Rack_2940_offset"].Value = (0);
```

```
        }

        if (Orig_Track_Length == 11760)
        {
            n = 5;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
```

```
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 13230)
        {
            n = 6;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = ((3.990 * obj_data[10, n]) + (3.360 *
obj_data[7,n]));
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
```

```
    component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_3990_offset"].Value = (0);
    component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
    component.Properties["Rail_Cover_2940_offset"].Value = (0);
    component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
    component.Properties["Rack_2940_offset"].Value = (0);
}

if (Orig_Track_Length == 14700)
{
    n = 7;
    component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
    component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
    component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
    component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
    component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
    component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
    component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
    component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
    component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
    component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
    component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
    component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
    component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
    component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
    component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

    component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
    component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
    component.Properties["Star_Rail_1470_dist"].Value = (1.470);
    component.Properties["Star_Rail_2310_dist"].Value = (2.310);
    component.Properties["Star_Rail_2730_dist"].Value = (2.730);
    component.Properties["Star_Rail_2940_dist"].Value = (2.940);
    component.Properties["Star_Rail_3150_dist"].Value = (3.150);
    component.Properties["Star_Rail_3360_dist"].Value = (3.360);
    component.Properties["Star_Rail_3570_dist"].Value = (3.570);
    component.Properties["Star_Rail_3780_dist"].Value = (3.780);
    component.Properties["Star_Rail_3990_dist"].Value = (3.990);
    component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
    component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
    component.Properties["Rack_1470_dist"].Value = (1.470);
    component.Properties["Rack_2940_dist"].Value = (2.940);
```

```
            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 16170)
        {
            n = 8;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
```

```
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 17640)
        {
            n = 9;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
```

```
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = ((3.990 * obj_data[10, n]) + (3.780 *
obj_data[9,n]));
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 19110)
        {
            n = 10;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);
```

```
            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = ((3.990 * obj_data[10, n]) + (2.940 *
obj_data[9,n]));
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 20580)
        {
            n = 11;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
```

```
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = ((3.990 * obj_data[10, n]) + (2.73 *
obj_data[4,n]));
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 22050)
        {
            n = 12;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
```

```
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2310_offset"].Value = ((3.990 * obj_data[10, n]) + (3.360 *
obj_data[7,n]));
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 23520)
```

```
{
    n = 13;
    component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
    component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
    component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
    component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
    component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
    component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
    component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
    component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
    component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
    component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
    component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
    component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
    component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
    component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
    component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

    component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
    component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
    component.Properties["Star_Rail_1470_dist"].Value = (1.470);
    component.Properties["Star_Rail_2310_dist"].Value = (2.310);
    component.Properties["Star_Rail_2730_dist"].Value = (2.730);
    component.Properties["Star_Rail_2940_dist"].Value = (2.940);
    component.Properties["Star_Rail_3150_dist"].Value = (3.150);
    component.Properties["Star_Rail_3360_dist"].Value = (3.360);
    component.Properties["Star_Rail_3570_dist"].Value = (3.570);
    component.Properties["Star_Rail_3780_dist"].Value = (3.780);
    component.Properties["Star_Rail_3990_dist"].Value = (3.990);
    component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
    component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
    component.Properties["Rack_1470_dist"].Value = (1.470);
    component.Properties["Rack_2940_dist"].Value = (2.940);

    component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
    component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
    component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
    component.Properties["Star_Rail_3990_offset"].Value = (0);
    component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
    component.Properties["Rail_Cover_2940_offset"].Value = (0);
```

```
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 24990)
        {
            n = 14;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = ((3.990 * obj_data[10, n]) + (3.150 *
obj_data[6,n]));
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
```

```
        component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
        component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
        component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
        component.Properties["Star_Rail_3990_offset"].Value = (0);
        component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
        component.Properties["Rail_Cover_2940_offset"].Value = (0);
        component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
        component.Properties["Rack_2940_offset"].Value = (0);
    }

    if (Orig_Track_Length == 26460)
    {
        n = 15;
        component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
        component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
        component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
        component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
        component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
        component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
        component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
        component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
        component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
        component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
        component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
        component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
        component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
        component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
        component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

        component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
        component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
        component.Properties["Star_Rail_1470_dist"].Value = (1.470);
        component.Properties["Star_Rail_2310_dist"].Value = (2.310);
        component.Properties["Star_Rail_2730_dist"].Value = (2.730);
        component.Properties["Star_Rail_2940_dist"].Value = (2.940);
        component.Properties["Star_Rail_3150_dist"].Value = (3.150);
        component.Properties["Star_Rail_3360_dist"].Value = (3.360);
        component.Properties["Star_Rail_3570_dist"].Value = (3.570);
        component.Properties["Star_Rail_3780_dist"].Value = (3.780);
        component.Properties["Star_Rail_3990_dist"].Value = (3.990);
        component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
        component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
        component.Properties["Rack_1470_dist"].Value = (1.470);
        component.Properties["Rack_2940_dist"].Value = (2.940);

        component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
        component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
```

```
            component.Properties[“Star_Rail_1470_offset”].Value = (3.990 * obj_data[10, n]);
            component.Properties[“Star_Rail_2310_offset”].Value = (3.990 * obj_data[10, n]);
            component.Properties[“Star_Rail_2730_offset”].Value = (3.990 * obj_data[10, n]);
            component.Properties[“Star_Rail_2940_offset”].Value = ((3.990 * obj_data[10, n]) + (3.150 *
obj_data[6,n]));
            component.Properties[“Star_Rail_3150_offset”].Value = (3.990 * obj_data[10, n]);
            component.Properties[“Star_Rail_3360_offset”].Value = (3.990 * obj_data[10, n]);
            component.Properties[“Star_Rail_3570_offset”].Value = (3.990 * obj_data[10, n]);
            component.Properties[“Star_Rail_3780_offset”].Value = (3.990 * obj_data[10, n]);
            component.Properties[“Star_Rail_3990_offset”].Value = (0);
            component.Properties[“Rail_Cover_1470_offset”].Value = (2.940 * obj_data[12, n]);
            component.Properties[“Rail_Cover_2940_offset”].Value = (0);
            component.Properties[“Rack_1470_offset”].Value = (2.940 * obj_data[14, n]);
            component.Properties[“Rack_2940_offset”].Value = (0);
        }

        if (Orig_Track_Length == 27930)
        {
            n = 16;
            component.Properties[“Rail_Weldment_Section_2940_count”].Value = (obj_data[0, n]);
            component.Properties[“Rail_Weldment_Section_4410_count”].Value = (obj_data[1, n]);
            component.Properties[“Star_Rail_1470_count”].Value = (obj_data[2, n]);
            component.Properties[“Star_Rail_2310_count”].Value = (obj_data[3, n]);
            component.Properties[“Star_Rail_2730_count”].Value = (obj_data[4, n]);
            component.Properties[“Star_Rail_2940_count”].Value = (obj_data[5, n]);
            component.Properties[“Star_Rail_3150_count”].Value = (obj_data[6, n]);
            component.Properties[“Star_Rail_3360_count”].Value = (obj_data[7, n]);
            component.Properties[“Star_Rail_3570_count”].Value = (obj_data[8, n]);
            component.Properties[“Star_Rail_3780_count”].Value = (obj_data[9, n]);
            component.Properties[“Star_Rail_3990_count”].Value = (obj_data[10, n]);
            component.Properties[“Rail_Cover_1470_count”].Value = (obj_data[11, n]);
            component.Properties[“Rail_Cover_2940_count”].Value = (obj_data[12, n]);
            component.Properties[“Rack_1470_count”].Value = (obj_data[13, n]);
            component.Properties[“Rack_2940_count”].Value = (obj_data[14, n]);

            component.Properties[“Rail_Weldment_Section_2940_dist”].Value = (2.940);
            component.Properties[“Rail_Weldment_Section_4410_dist”].Value = (4.410);
            component.Properties[“Star_Rail_1470_dist”].Value = (1.470);
            component.Properties[“Star_Rail_2310_dist”].Value = (2.310);
            component.Properties[“Star_Rail_2730_dist”].Value = (2.730);
            component.Properties[“Star_Rail_2940_dist”].Value = (2.940);
            component.Properties[“Star_Rail_3150_dist”].Value = (3.150);
            component.Properties[“Star_Rail_3360_dist”].Value = (3.360);
            component.Properties[“Star_Rail_3570_dist”].Value = (3.570);
            component.Properties[“Star_Rail_3780_dist”].Value = (3.780);
            component.Properties[“Star_Rail_3990_dist”].Value = (3.990);
            component.Properties[“Rail_Cover_1470_dist”].Value = (1.470);
            component.Properties[“Rail_Cover_2940_dist”].Value = (2.940);
```

```
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 29400)
        {
            n = 17;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
```

```
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = ((3.990 * obj_data[10, n]) + (1.470 *
obj_data[2,n]));
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 30870)
        {
            n = 18;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
```

```
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2310_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2730_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2940_offset"].Value = ((3.990 * obj_data[10, n]) + (3.570 *
obj_data[8,n]));
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }

        if (Orig_Track_Length == 32340)
        {
            n = 19;
            component.Properties["Rail_Weldment_Section_2940_count"].Value = (obj_data[0, n]);
            component.Properties["Rail_Weldment_Section_4410_count"].Value = (obj_data[1, n]);
            component.Properties["Star_Rail_1470_count"].Value = (obj_data[2, n]);
            component.Properties["Star_Rail_2310_count"].Value = (obj_data[3, n]);
            component.Properties["Star_Rail_2730_count"].Value = (obj_data[4, n]);
            component.Properties["Star_Rail_2940_count"].Value = (obj_data[5, n]);
            component.Properties["Star_Rail_3150_count"].Value = (obj_data[6, n]);
            component.Properties["Star_Rail_3360_count"].Value = (obj_data[7, n]);
            component.Properties["Star_Rail_3570_count"].Value = (obj_data[8, n]);
            component.Properties["Star_Rail_3780_count"].Value = (obj_data[9, n]);
            component.Properties["Star_Rail_3990_count"].Value = (obj_data[10, n]);
            component.Properties["Rail_Cover_1470_count"].Value = (obj_data[11, n]);
```

```
            component.Properties["Rail_Cover_2940_count"].Value = (obj_data[12, n]);
            component.Properties["Rack_1470_count"].Value = (obj_data[13, n]);
            component.Properties["Rack_2940_count"].Value = (obj_data[14, n]);

            component.Properties["Rail_Weldment_Section_2940_dist"].Value = (2.940);
            component.Properties["Rail_Weldment_Section_4410_dist"].Value = (4.410);
            component.Properties["Star_Rail_1470_dist"].Value = (1.470);
            component.Properties["Star_Rail_2310_dist"].Value = (2.310);
            component.Properties["Star_Rail_2730_dist"].Value = (2.730);
            component.Properties["Star_Rail_2940_dist"].Value = (2.940);
            component.Properties["Star_Rail_3150_dist"].Value = (3.150);
            component.Properties["Star_Rail_3360_dist"].Value = (3.360);
            component.Properties["Star_Rail_3570_dist"].Value = (3.570);
            component.Properties["Star_Rail_3780_dist"].Value = (3.780);
            component.Properties["Star_Rail_3990_dist"].Value = (3.990);
            component.Properties["Rail_Cover_1470_dist"].Value = (1.470);
            component.Properties["Rail_Cover_2940_dist"].Value = (2.940);
            component.Properties["Rack_1470_dist"].Value = (1.470);
            component.Properties["Rack_2940_dist"].Value = (2.940);

            component.Properties["Rail_Weldment_Section_2940_offset"].Value = (4.410 * obj_data[1,
n]);
            component.Properties["Rail_Weldment_Section_4410_offset"].Value = (0);
            component.Properties["Star_Rail_1470_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_2310_offset"].Value = ((3.990 * obj_data[10, n]) + (2.940 *
obj_data[5,n]) + (2.73 * obj_data[4,n]));
            component.Properties["Star_Rail_2730_offset"].Value = ((3.990 * obj_data[10, n]) + (2.940 *
obj_data[5,n]));
            component.Properties["Star_Rail_2940_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3150_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3360_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3570_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3780_offset"].Value = (3.990 * obj_data[10, n]);
            component.Properties["Star_Rail_3990_offset"].Value = (0);
            component.Properties["Rail_Cover_1470_offset"].Value = (2.940 * obj_data[12, n]);
            component.Properties["Rail_Cover_2940_offset"].Value = (0);
            component.Properties["Rack_1470_offset"].Value = (2.940 * obj_data[14, n]);
            component.Properties["Rack_2940_offset"].Value = (0);
        }
      }
    }
  }
}
```

XML

```
<?xml version=”1.0” encoding=”utf-8” ?>
<lc:LibraryCompiler xmlns:lc=”urn:abb-robotics-robotstudio-librarycompiler”
                                                                    xmlns=”urn:abb-
robotics-robotstudio-graphiccomponent”>
        <lc:Library fileName=”Track_Builder.rslib”>
                <lc:DocumentProperties>
                        <lc:Author>jkopacz</lc:Author>
                        <lc:Image source=”Track_Builder.png”/>
                </lc:DocumentProperties>
                <SmartComponent name=”Track_Builder” icon=”Track_Builder.png”

        codeBehind=”Track_Builder.CodeBehind,Track_Builder.dll”

        canBeSimulated=”false”>
                        <Properties>
    <!—Input Variables→
    <DynamicProperty name=”Track_Length_mm” valueType=”System.Int32” value=”4410”>
     <Attribute key=”Allowed”
value=”4410;5880;7350;8820;10290;11760;13230;14700;16170;17640;19110;20580;22050;23520;2499
0;26460;27930;29400;30870;32340”/>
    </DynamicProperty>
    <DynamicProperty name=”Gantry” valueType=”System.String” value=”None”>
     <Attribute key=”Allowed” value=”Gantry_2500H; Gantry_3500H; Gantry_4000H;
Inv_Gantry_2500H; Inv_Gantry_3500H; Inv_Gantry_4000H; None”/>
    </DynamicProperty>
    <DynamicProperty name=”Boom” valueType=”System.String” value=”None”>
     <Attribute key=”Allowed” value=”Boom_1500; Boom_2000; Boom_2500; Boom_3000; None”/>
    </DynamicProperty>
    <DynamicProperty name=”RTT” valueType=”System.String” value=”None”>
     <Attribute key=”Allowed” value=”Robot; None”/>
    </DynamicProperty>

    <DynamicProperty name=”Water_Cooler” valueType=”System.String” value=”None”>
     <Attribute key=”Allowed” value=”Lincoln;Miller”/>
    </DynamicProperty>
    <DynamicProperty name=”Wire_Barrel” valueType=”System.String” value=”None”>
     <Attribute key=”Allowed”
value=”Single_Wire_Barrel_Low;Tandum_Wire_Barrel_Low;Wire_Barrel_High”/>
    </DynamicProperty>
    <DynamicProperty name=”Torch_Cleaner_Bullseye” valueType=”System.String” value=”None”>
     <Attribute key=”Allowed” value=”yes;no”/>
    </DynamicProperty>

    <!—Output Variables Rail Weldment→
    <DynamicProperty name=”Rail_Weldment_Section_2940_count” valueType=”System.Double”
value=”0” readOnly=”true”>
     <Attribute key=”MinValue” value=”0”/>
```

```
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Rail_Weldment_Section_4410_count" valueType="System.Double"
value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <!—Output Variables Star Rail→
    <DynamicProperty name="Star_Rail_1470_count" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_2310_count" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_2730_count" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_2940_count" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3150_count" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3360_count" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3570_count" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3780_count" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
```

```
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3990_count" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <!—Output Variables Rail Cover→
    <DynamicProperty name="Rail_Cover_1470_count" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Rail_Cover_2940_count" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <!—Output Variables Rack→
    <DynamicProperty name="Rack_1470_count" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Rack_2940_count" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <!—Output Variables Rail Weldment Distance→
    <DynamicProperty name="Rail_Weldment_Section_2940_dist" valueType="System.Double"
value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Rail_Weldment_Section_4410_dist" valueType="System.Double"
value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <!—Output Variables Star Rail Distance→
    <DynamicProperty name="Star_Rail_1470_dist" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
```

```
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_2310_dist" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_2730_dist" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_2940_dist" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3150_dist" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3360_dist" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3570_dist" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3780_dist" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3990_dist" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <!—Output Variables Rail Cover Distance→
    <DynamicProperty name="Rail_Cover_1470_dist" valueType="System.Double" value="0"
readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
```

```
    <DynamicProperty name="Rail_Cover_2940_dist" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <!—Output Variables Rack Distance→
    <DynamicProperty name="Rack_1470_dist" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Rack_2940_dist" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <!—Output Variables Rail Weldment Offset→
    <DynamicProperty name="Rail_Weldment_Section_2940_offset" valueType="System.Double"
value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Rail_Weldment_Section_4410_offset" valueType="System.Double"
value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <!—Output Variables Star Rail Offset→
    <DynamicProperty name="Star_Rail_1470_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_2310_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_2730_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_2940_offset" valueType="System.Double" value="0"
readOnly="true">
```

```
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3150_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3360_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3570_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3780_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Star_Rail_3990_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <!—Output Variables Rail Cover Offset→
    <DynamicProperty name="Rail_Cover_1470_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Rail_Cover_2940_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>

    <!—Output Variables Rack Offset→
    <DynamicProperty name="Rack_1470_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
```

```
    <DynamicProperty name="Rack_2940_offset" valueType="System.Double" value="0"
readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
                        </Properties>
                        <Bindings>
                        </Bindings>
                        <GraphicComponents>
                        </GraphicComponents>
                        <Assets>
                                <Asset source="Track_Builder.dll"/>
                        </Assets>
                </SmartComponent>
        </lc:Library>
</lc:LibraryCompiler>
```

Segment Order

SYSTEM #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TOTAL RAIL LENGTH	4410	5880	7350	8820	10290	11760	13230	14700	16170	17640	19110	20580	22050	23520	24990	26460	27930	29400	30870	32340
TOTAL LENGTH - (STOPS+LIMIT CAMS)	3730	5200	6670	8140	9610	11080	12550	14020	15490	16960	18430	19900	21370	22840	24310	25780	27250	28720	30190	31660
TOTAL TRAVEL LENGTH, MM	2925	4395	5865	7335	8805	10275	11745	13215	14685	16155	17625	19095	20565	22035	23505	24975	26445	27915	29385	30855
TOTAL TRAVEL LENGTH, M	2.93	4.40	5.87	7.34	8.81	10.28	11.75	13.22	14.69	16.16	17.63	19.10	20.57	22.04	23.51	24.98	26.45	27.92	29.39	30.86
SUM OF RAIL WELDMENTS	4410	5880	7350	8820	10290	11760	13230	14700	16170	17640	19110	20580	22050	23520	24990	26460	27930	29400	30870	32340
SECTION 1	4410	2940	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410
SECTION 2		2940	2940	4410	2940	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410
SECTION 3					2940	2940	4410	2940	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410	4410
SECTION 4								2940	2940	4410	2940	4410	4410	4410	4410	4410	4410	4410	4410	4410
SECTION 5											2940	2940	4410	2940	4410	4410	4410	4410	4410	4410
SECTION 6														2940	2940	4410	2940	4410	4410	4410
SECTION 7																	2940	2940	4410	2940
SECTION 8																				2940
SECTION 9																				
SECTION 10																				
SECTION 11																				
SECTION 12																				
SECTION 13																				
SECTION 14																				
SECTION 15																				
2940.00									RAIL WELDMENT											
4410.00																				

SYSTEM #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TOTAL RAIL LENGTH	4410	5880	7350	8820	10290	11760	13230	14700	16170	17640	19110	20580	22050	23520	24990	26460	27930	29400	30870	32340
TOTAL TRAVEL LENGTH, MM	2925	4395	5865	7335	8805	10275	11745	13215	14685	16155	17625	19095	20565	22035	23505	24975	26445	27915	29385	30855
TOTAL TRAVEL LENGTH, MM	2.93	4.40	5.87	7.34	8.81	10.28	11.75	13.22	14.69	16.16	17.63	19.10	20.57	22.04	23.51	24.98	26.45	27.92	29.39	30.86
TOTAL MINUS STOPS	3990	5460	6930	8400	9870	11340	12810	14280	15750	17220	18690	20160	21630	23100	24570	26040	27510	28980	30450	31920
SUM OF RAIL SECTIONS	3990	5460	6930	8400	9870	11340	12810	14280	15750	17220	18690	20160	21630	23100	24570	26040	27510	28980	30450	31920
SECTION 1	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990
SECTION 2		1470	2940	2940	2940	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990
SECTION 3				1470	2940	3360	3360	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990	3990
SECTION 4							1470	2310	3780	3780	3780	3990	3990	3990	3990	3990	3990	3990	3990	3990
SECTION 5										1470	2940	2730	3360	3990	3990	3990	3990	3990	3990	3990
SECTION 6												1470	2310	3150	3150	3150	3990	3990	3990	3990
SECTION 7															1470	2940	3570	3570	3570	2940
SECTION 8																		1470	2940	2730
SECTION 9																				2310

1470
2310
2730
2940
3150
3360
3570
3780

STAR RAIL

SYSTEM #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TOTAL RAIL LENGTH	4410	5880	7350	8820	10290	11760	13230	14700	16170	17640	19110	20580	22050	23520	24990	26460	27930	29400	30870	32340
SUM OF COVER SECTIONS	4410	5880	7350	8820	10290	11760	13230	14700	16170	17640	19110	20580	22050	23520	24990	26460	27930	29400	30870	32340
SECTION 1	1470	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940
SECTION 2	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940
SECTION 3			1470	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940
SECTION 4					1470	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940
SECTION 5							1470	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940
SECTION 6									1470	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940	2940
SECTION 7											1470	2940	2940	2940	2940	2940	2940	2940	2940	2940
SECTION 8													1470	2940	2940	2940	2940	2940	2940	2940
SECTION 9															1470	2940	2940	2940	2940	2940
SECTION 10																	1470	2940	2940	2940
SECTION 11																			1470	2940

1470 MM (57.87 IN)
2940 MM(115.75 IN)
4410 MM(173.62 IN)

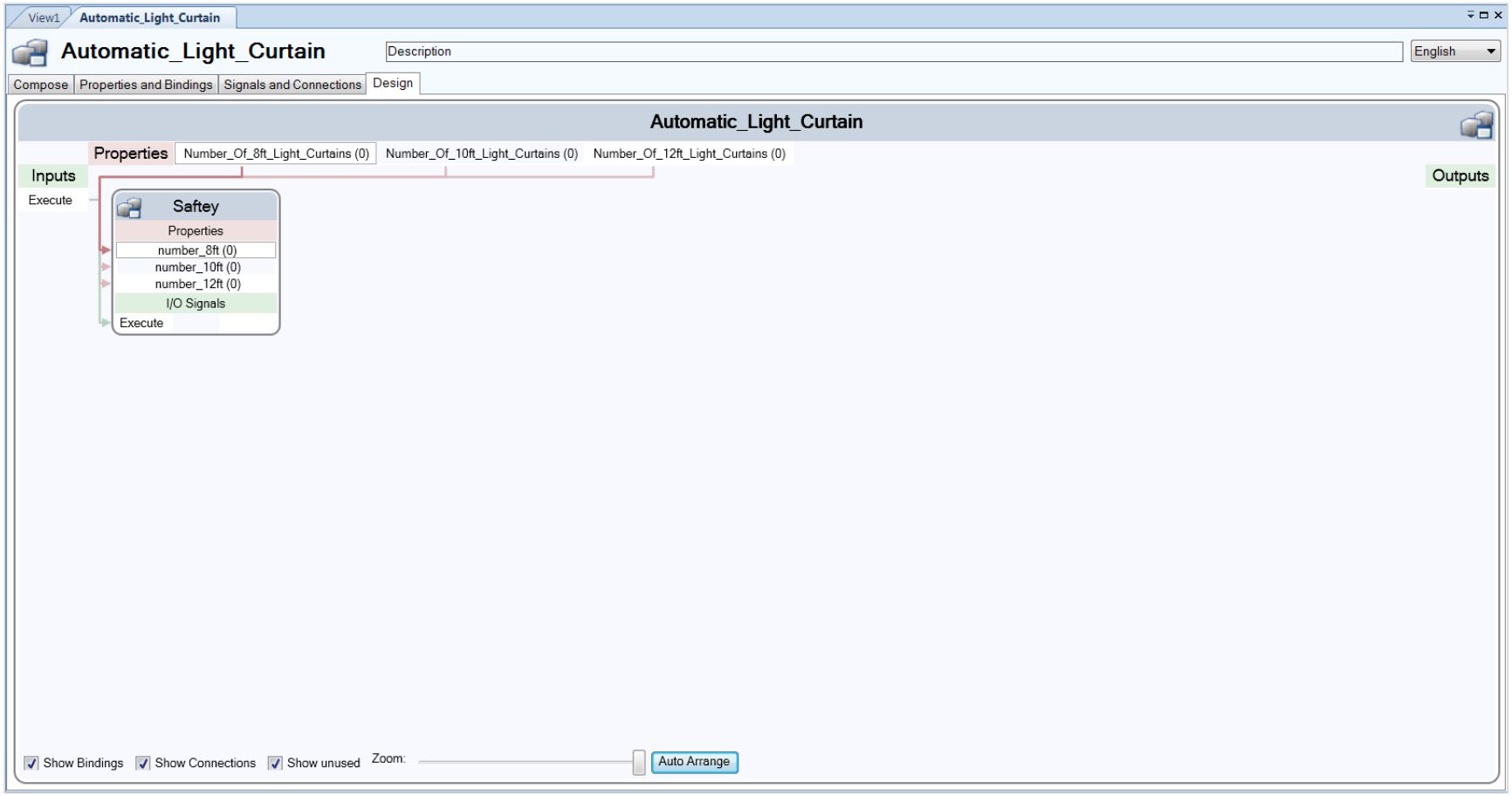
RAIL COVER

SYSTEM #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TOTAL RAIL LENGTH	4410	5880	7350	8820	10290	11760	13230	14700	16170	17640	19110	20580	22050	23520	24990	26460	27930	29400	30870	32340
SUM OF RACK SECTIONS	4408.530724	5878.796086	7349.061448	8819.32681	10289.59217	11759.85753	13230.1229	14700.38826	16170.65362	17640.65362	19111.18434	20581.44971	22051.71507	23521.98043	24992.24579	26462.51115	27932.77652	29403.04188	30873.30724	32343.5726
SECTION 1	2940.530724	1470.265362	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724
SECTION 2	1468	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724
SECTION 3		1468	1468	1470.265362	2940.530724	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362
SECTION 4				1468	1468	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724
SECTION 5						1468	1470.265362	2940.530724	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362
SECTION 6							1468	1468	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724
SECTION 7									1468	1470	2940.530724	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362	1470.265362
SECTION 8										1468	1468	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724
SECTION 9												1468	1470.265362	2940.530724	1470.265362	2940.530724	1470.265362	1470.265362	1470.265362	1470.265362
SECTION 10													1468	1468	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724	2940.530724
SECTION 11															1468	1468	2940.530724	2940.530724	1470.265362	1470.265362
SECTION 12																	1468	1470.265362	2940.530724	2940.530724
SECTION 13																		1468	1470.265362	2940.530724
SECTION 14																			1468	1468
SECTION 15																				

1470.265362	RACK
2940.530724	
1468 (NO RACK)	

Light_Curtain

RobotStudio



CodeBehind

```
using System;
using System.Collections.Generic;
using System.Text;

using ABB.Robotics.Math;
using ABB.Robotics.RobotStudio;
using ABB.Robotics.RobotStudio.Stations;

namespace Saftey
{
   public class CodeBehind : SmartComponentCodeBehind
   {
      public override void OnIOSignalValueChanged(SmartComponent component, IOSignal
changedSignal)
      {
         //Get variables
         int number_8ft = (int)component.Properties["number_8ft"].Value;
         int number_10ft = (int)component.Properties["number_10ft"].Value;
         int number_12ft = (int)component.Properties["number_12ft"].Value;
         int count;

         //Open Active Station
         Station station = Project.ActiveProject as Station;
         GraphicComponent temp;

         //Reset Count and Create New String
         count = 0;
         System.Text.StringBuilder sb = new System.Text.StringBuilder("Curtain_8ft_0");
         System.Text.StringBuilder sb_light = new System.Text.StringBuilder("Light_8ft_0");

         while (count < (number_8ft))
         {
            //Create Correct Name
            sb.Remove(12, 1);
            sb_light.Remove(10,1);
            sb.Append(count);
            sb_light.Append(count);

            //Check For Light Curtain with Current Name, Create One if False
            station.GraphicComponents.TryGetGraphicComponent(sb.ToString(), out temp);
            if (temp == null)
            {
               // Load Mechanism Curtain
               GraphicComponentLibrary mylib;
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Light_Curtain\\WolfLightCurtain_8ft.rslib", true);
```

```
    Mechanism Curtain = (Mechanism)mylib.RootComponent.CopyInstance();
    Curtain.Name = sb.ToString();
    station.GraphicComponents.Add(Curtain);
    Curtain.DisconnectFromLibrary();

    //Get Current Joint Values
    double[] CurtainLength;
    CurtainLength = Curtain.GetJointValues();
    Curtain.Frames.ToArray();

    //Create Light Curtain
    Part Light = new Part();
    Light.Name = sb_light.ToString();

    //Move Light Curtain to Correct Location
    Matrix4 rot = Curtain.Transform.GlobalMatrix;
    Vector3 size = new Vector3(CurtainLength[0] + .13096, .03048, 1.20371);

    //Create Body and set Color and Transparency
    Body square = Body.CreateSolidBox(rot, size);
    square.Transform.Z = .33439;
    square.Name = "box";
    Light.Bodies.Add(square);
    VSTABridge.SetColor(Light, 255, 0, 0, 126);

    //Attach box to base of mechanism
    GraphicComponent home;
    Curtain.GetParentLink(0, out home);
    Part home1 = (Part)home;
    home1.Attach(Light, false, new Matrix4(new Vector3(0, 0, 0)));
}
else
{
    //Get Current Joint Values
    Mechanism Curtain = (Mechanism)temp;
    double[] CurtainLength;
    CurtainLength = Curtain.GetJointValues();

    //Delete Old Light
    GraphicComponent to_del;
    station.GraphicComponents.TryGetGraphicComponent(sb_light.ToString(), out to_del);
    if (to_del != null)
    {
        station.GraphicComponents.Remove(to_del);
    }

    //Create Light Curtain
    Part Light = new Part();
```

```
      Light.Name = sb_light.ToString();

      //Move Light Curtain to Correct Location
      station.GraphicComponents.Add(Light);
      Matrix4 rot = Curtain.Transform.GlobalMatrix;

      //Create Body and Set Color and Transparency
      Vector3 size = new Vector3(CurtainLength[0] + .13096, .03048, 1.20371);
      Body square = Body.CreateSolidBox(rot, size);
      square.Transform.Z = .33439;
      square.Name = "box";
      Light.Bodies.Add(square);
      VSTABridge.SetColor(Light, 255, 0, 0, 126);

      //Attach box to base of mechanism
      GraphicComponent home;
      Curtain.GetParentLink(0, out home);
      Part home1 = (Part)home;
      home1.Attach(Light, false, new Matrix4(new Vector3(0, 0, 0)));
    }
    count = count + 1;
}

//Delete undesired
while (count < 11)
{
    //Create Correct Name
    sb.Remove(12, 1);
    sb_light.Remove(10, 1);
    sb.Append(count);
    sb_light.Append(count);
    GraphicComponent to_del;

    //Delete Old Light
    station.GraphicComponents.TryGetGraphicComponent(sb_light.ToString(), out to_del);
    if (to_del != null)
    {
      station.GraphicComponents.Remove(to_del);
    }

    //Delete Old Curtain
    station.GraphicComponents.TryGetGraphicComponent(sb.ToString(), out to_del);
    if (to_del != null)
    {
      station.GraphicComponents.Remove(to_del);
    }
    count = count + 1;
}
```

```
        //Reset Count and Create New String
        count = 0;
        System.Text.StringBuilder sb2 = new System.Text.StringBuilder("Curtain_10ft_0");
        System.Text.StringBuilder sb2_light = new System.Text.StringBuilder("Light_10ft_0");

        while (count < (number_10ft))
        {
          //Create Correct String
          sb2.Remove(13, 1);
          sb2_light.Remove(11, 1);
          sb2.Append(count);
          sb2_light.Append(count);

          //Check For Light Curtain with Current Name, Create One if False
          station.GraphicComponents.TryGetGraphicComponent(sb2.ToString(), out temp);
          if (temp == null)
          {
            // Load Mechanism Curtain
            GraphicComponentLibrary mylib;
            mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Light_Curtain\\WolfLightCurtain_10ft.rslib", true);
            Mechanism Curtain = (Mechanism)mylib.RootComponent.CopyInstance();
            Curtain.Name = sb2.ToString();
            station.GraphicComponents.Add(Curtain);
            Curtain.DisconnectFromLibrary();

            //Get Current Joint Values
            double[] CurtainLength;
            CurtainLength = Curtain.GetJointValues();
            Curtain.Frames.ToArray();

            //Create Light Curtain
            Part Light = new Part();
            Light.Name = sb2_light.ToString();

            //Move Light Curtain to Correct Location
            Matrix4 rot = Curtain.Transform.GlobalMatrix;
            Vector3 size = new Vector3(CurtainLength[0] + .13096, .03048, 1.20371);

            //Create Body and Set Color and Transparency
            Body square = Body.CreateSolidBox(rot, size);
            square.Transform.Z = .33439;
            square.Name = "box";
            Light.Bodies.Add(square);
            VSTABridge.SetColor(Light, 255, 0, 0, 126);

            //Attach box to base of mechanism
```

```
        GraphicComponent home;
        Curtain.GetParentLink(0, out home);
        Part home1 = (Part)home;
        home1.Attach(Light, false, new Matrix4(new Vector3(0, 0, 0)));
    }
    else
    {
        //Get Current Joint Values
        Mechanism Curtain = (Mechanism)temp;
        double[] CurtainLength;
        CurtainLength = Curtain.GetJointValues();

        //Delete Old Light
        GraphicComponent to_del;
        station.GraphicComponents.TryGetGraphicComponent(sb2_light.ToString(), out to_del);
        if (to_del != null)
        {
            station.GraphicComponents.Remove(to_del);
        }

        //Create Light Curtain
        Part Light = new Part();
        Light.Name = sb2_light.ToString();
        station.GraphicComponents.Add(Light);
        Matrix4 rot = Curtain.Transform.GlobalMatrix;

        //Create Body and Set Color and Transparency
        Vector3 size = new Vector3(CurtainLength[0] + .13096, .03048, 1.20371);
        //Body square = Body.CreateSolidBox(origin, size);
        Body square = Body.CreateSolidBox(rot, size);
        square.Transform.Z = .33439;
        square.Name = "box";
        Light.Bodies.Add(square);
        VSTABridge.SetColor(Light, 255, 0, 0, 126);

        //Attach box to base of mechanism
        GraphicComponent home;
        Curtain.GetParentLink(0, out home);
        Part home1 = (Part)home;
        home1.Attach(Light, false, new Matrix4(new Vector3(0, 0, 0)));
    }
    count = count + 1;
}

//Delete undesired
while (count < 11)
{
    //Create Correct Name
```

```
            sb2.Remove(13, 1);
            sb2_light.Remove(11, 1);
            sb2.Append(count);
            sb2_light.Append(count);
            GraphicComponent to_del;

            //Delete Old Light
            station.GraphicComponents.TryGetGraphicComponent(sb2_light.ToString(), out to_del);
            if (to_del != null)
            {
               station.GraphicComponents.Remove(to_del);
            }

            //Delete Old Curtain
            station.GraphicComponents.TryGetGraphicComponent(sb2.ToString(), out to_del);
            if (to_del != null)
            {
               station.GraphicComponents.Remove(to_del);
            }
            count = count + 1;
         }

         //Reset Count and Create New String
         count = 0;
         System.Text.StringBuilder sb3 = new System.Text.StringBuilder("Curtain_12ft_0");
         System.Text.StringBuilder sb3_light = new System.Text.StringBuilder("Light_12ft_0");

         while (count < (number_12ft))
         {
            //Create Correct Name
            sb3.Remove(13, 1);
            sb3_light.Remove(11, 1);
            sb3.Append(count);
            sb3_light.Append(count);

            //Check For Light Curtain with Current Name, Create One if False
            station.GraphicComponents.TryGetGraphicComponent( sb3.ToString(), out temp);
            if (temp == null)
            {
               // Load Mechanism Curtain
               GraphicComponentLibrary mylib;
               mylib = GraphicComponentLibrary.Load("S:\\Library
Build\\Custom_Lib\\Light_Curtain\\WolfLightCurtain_12ft.rslib", true);
               Mechanism Curtain = (Mechanism)mylib.RootComponent.CopyInstance();
               Curtain.Name = sb3.ToString();
               station.GraphicComponents.Add(Curtain);
               Curtain.DisconnectFromLibrary();
```

```
    //Get Joint Values
    double[] CurtainLength;
    CurtainLength = Curtain.GetJointValues();
    Curtain.Frames.ToArray();

    //Create Light Curtain
    Part Light = new Part();
    Light.Name = sb3_light.ToString();

    //Move Light Curtain to Correct Location
    Matrix4 rot = Curtain.Transform.GlobalMatrix;
    Vector3 size = new Vector3(CurtainLength[0] + .13096, .03048, 1.20371);

    //Create Body and Set Color and Transparency
    Body square = Body.CreateSolidBox(rot, size);
    square.Transform.Z = .33439;
    square.Name = "box";
    Light.Bodies.Add(square);
    VSTABridge.SetColor(Light, 255, 0, 0, 126);

    //Attach box to base of mechanism
    GraphicComponent home;
    Curtain.GetParentLink(0, out home);
    Part home1 = (Part)home;
    home1.Attach(Light, false, new Matrix4(new Vector3(0, 0, 0)));
}
else
{

    //Get Current Joint Values
    Mechanism Curtain = (Mechanism)temp;
    double[] CurtainLength;
    CurtainLength = Curtain.GetJointValues();

    //Delete Old Light
    GraphicComponent to_del;
    station.GraphicComponents.TryGetGraphicComponent(sb3_light.ToString(), out to_del);
    if (to_del != null)
    {
       station.GraphicComponents.Remove(to_del);
    }

    //Create Light Curtain
    Part Light = new Part();
    Light.Name = sb3_light.ToString();
    station.GraphicComponents.Add(Light);
    Matrix4 rot = Curtain.Transform.GlobalMatrix;
```

```
            //Create Body and Set Color and Transparency
            Vector3 size = new Vector3(CurtainLength[0] + .13096, .03048, 1.20371);
            Body square = Body.CreateSolidBox(rot, size);
            square.Transform.Z = .33439;
            square.Name = "box";
            Light.Bodies.Add(square);
            VSTABridge.SetColor(Light, 255, 0, 0, 126);

            //Attach box to base of mechanism
            GraphicComponent home;
            Curtain.GetParentLink(0, out home);
            Part home1 = (Part)home;
            home1.Attach(Light, false, new Matrix4(new Vector3(0, 0, 0)));
          }
          count = count + 1;
        }

        //Delete undesired
        while (count < 11)
        {

          //Create Correct Name
          sb3.Remove(13, 1);
          sb3_light.Remove(11, 1);
          sb3.Append(count);
          sb3_light.Append(count);
          GraphicComponent to_del;

          //Delete Old Light
          station.GraphicComponents.TryGetGraphicComponent(sb3_light.ToString(), out to_del);
          if (to_del != null)
          {
            station.GraphicComponents.Remove(to_del);
          }

          //Delete Old Curtain
          station.GraphicComponents.TryGetGraphicComponent(sb3.ToString(), out to_del);
          if (to_del != null)
          {
            station.GraphicComponents.Remove(to_del);
          }
          count = count + 1;
        }
      }
    }
}
```

XML

```
<?xml version=”1.0” encoding=”utf-8” ?>
<lc:LibraryCompiler xmlns:lc=”urn:abb-robotics-robotstudio-librarycompiler”
                                                                    xmlns=”urn:abb-
robotics-robotstudio-graphiccomponent”>
        <lc:Library fileName=”Saftey.rslib”>
                <lc:DocumentProperties>
                        <lc:Author>jkopacz</lc:Author>
                        <lc:Image source=”Saftey.png”/>
                </lc:DocumentProperties>
                <SmartComponent name=”Saftey” icon=”Saftey.png”

        codeBehind=”Saftey.CodeBehind,Saftey.dll”

        canBeSimulated=”false”>
                        <Properties>
    <!—Create Robot Studio Input Variables→
    <DynamicProperty name=”number_8ft” valueType=”System.Int32” value=”0”>
     <Attribute key=”Allowed” value=”0;1;2;3;4;5;6;7;8;9”/>
    </DynamicProperty>
    <DynamicProperty name=”number_10ft” valueType=”System.Int32” value=”0”>
     <Attribute key=”Allowed” value=”0;1;2;3;4;5;6;7;8;9”/>
    </DynamicProperty>
    <DynamicProperty name=”number_12ft” valueType=”System.Int32” value=”0”>
     <Attribute key=”Allowed” value=”0;1;2;3;4;5;6;7;8;9”/>
    </DynamicProperty>
                        </Properties>
                        <Bindings>
                        </Bindings>
                        <Signals>
                                <IOSignal name=”Execute” signalType=”DigitalInput”/>
                        </Signals>
                        <GraphicComponents>
                        </GraphicComponents>
                        <Assets>
                                <Asset source=”Saftey.dll”/>
                        </Assets>
                </SmartComponent>
        </lc:Library>
</lc:LibraryCompiler>
```

APPENDIX D. SOURCE CODE ADD-IN

Positioners

Class1

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;
using System.IO;
using System.Windows.Forms;

using ABB.Robotics.Math;
using ABB.Robotics.RobotStudio;
using ABB.Robotics.RobotStudio.Environment;
using ABB.Robotics.RobotStudio.Stations;


//        To activate this Add-In you have to install it in the RobotStudio Add-In directory,
//  typically C:\Program Files\Common Files\ABB Industrial IT\Robotics IT\RobotStudio\AddIns
//  Copy Wolf_Positioners_Addin.dll to the Add-In directory, and copy Wolf_Positioners_Addin.rsaddin
to the
//  Add-In directory.
//
//  The rsaddin file provides additionals information to the RobotStudio host before the
//  Add-In assembly has been loaded, as specified in RobotStudioAddin.xsd. It also
//  allows more flexibility in locating the Add-In assembly, which can be useful
//        during development.
namespace Wolf_Positioners_Addin
{
   public class Class1
   {
      public static class settings
      {
         public static string skyhook_capacity = "1000Kg";
         public static string skyhook_throw = "1500";
         public static string skyhook_drop = "500";

         public static string HSTS_capacity = "500";
         public static string HSTS_riser_height = "500";

         public static string DC_riser_height = "500";
      }
```

```
    static void button_ExecuteCommand(object sender, EventArgs e)
    {
       Form1 form = new Form1();
       form.Show();
    }

    static void button2_ExecuteCommand(object sender, EventArgs e)
    {
       Form2 form2 = new Form2();
       form2.Show();
    }

    static void button3_ExecuteCommand(object sender, EventArgs e)
    {
       Form5 form5 = new Form5();
       form5.Show();
    }

    static void button_SH_Lift_ExecuteCommand(object sender, EventArgs e)
    {
       Form3 form3 = new Form3();
       form3.Show();
    }

    static void DC_ExecuteCommand(object sender, EventArgs e)
    {
       Form4 form4 = new Form4();
       form4.Show();
    }

    static void Fence_ExecuteCommand(object sender, EventArgs e)
    {
       System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\Smart_Components\\");

       Station station = Project.ActiveProject as Station;
       GraphicComponentLibrary mylib;

       //Load Fence
       string Path_fence;
       Path_fence = file_base.ToString() + "Automatic_Fencing_Final.rslib";
       mylib = GraphicComponentLibrary.Load(Path_fence, true);
       station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());

    }

    static void Light_ExecuteCommand(object sender, EventArgs e)
    {
```

```
        System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\Smart_Components\\");

        Station station = Project.ActiveProject as Station;
        GraphicComponentLibrary mylib;

        //Load Fence
        string Path_light;
        Path_light = file_base.ToString() + "Automatic_Light_Curtain_Final.rslib";
        mylib = GraphicComponentLibrary.Load(Path_light, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
    }

    static void TT_ExecuteCommand(object sender, EventArgs e)
    {
        System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\Smart_Components\\");

        Station station = Project.ActiveProject as Station;
        GraphicComponentLibrary mylib;

        //Load Fence
        string Path_tt;
        Path_tt = file_base.ToString() + "Automatic_Track_Final.rslib";
        mylib = GraphicComponentLibrary.Load(Path_tt, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
    }

    public static void Build()
    {
        if (settings.skyhook_capacity == "1000Kg")
        {
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;

            System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\SkyHook\\1000Kg\\");

            //Load Base
            string Path_base;
            string name_base = "1000Kg_SH_Base";
            Path_base = file_base.ToString() + "1000Kg_SH_Base.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base;
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
```

```
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Arm
            String Path_arm;
            String name_arm = "1000Kg_SH_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
            Path_arm = file_base.ToString();
            Path_arm = Path_arm + "1000Kg_SH_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
            GraphicComponent Arm = null;

            if (File.Exists(Path_arm))
            {
               mylib = GraphicComponentLibrary.Load(Path_arm, true);
               station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
               station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
               Arm.DisconnectFromLibrary();
            }
            else
            {
               Logger.AddMessage(new LogMessage("This is not a valid selection"));
               Logger.AddMessage(new LogMessage("For a 1000Kg capacity the options are as follows"));
               Logger.AddMessage(new LogMessage("Throw: 400, 500, 600, 700, 800, 900, 1000, 1100,
1200 "));
               Logger.AddMessage(new LogMessage("Drop: 400, 500, 600, 700, 800, 900, 1000"));
               return;
            }

            //Load Platter
            String Path_Platter;
            String name_Platter = "1000Kg_SH_Platter";
            Path_Platter = file_base.Append("1000Kg_SH_Platter.rslib").ToString();

            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "SkyHook";
            mb.AddLink("Base", Base);
            mb.AddLink("Link1", Arm);
            mb.AddLink("Link2", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Link1", new Vector3(1, 0, (Convert.ToInt32(settings.skyhook_drop)
* .001)), new Vector3(0, 0, (Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational,
true);
```

```
            mb.AddJoint("J2", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
            mb.SetJointLimit("J1", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
            mb.SetJointLimit("J2", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "SkyHook";

            //Load Riser
            if ((Convert.ToInt32(settings.skyhook_drop) + 800) > Convert.ToInt32(settings.skyhook_throw))
            {
               string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU5000Kg Risers\\";
               mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU5000_" +
Convert.ToString(Convert.ToInt32(settings.skyhook_drop) + 800) + "H.rslib", true);
               Part Riser = (Part)mylib.RootComponent.CopyInstance();
               Riser.Name = "Riser";
               station.GraphicComponents.Add(Riser);
               Riser.Transform.RY = -90 * (Math.PI / 180);
               Riser.Transform.RZ = -180 * (Math.PI / 180);
               Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
               Riser.Transform.X = (-1811.439 - (Convert.ToInt32(settings.skyhook_throw) - 1500)) * .001;

               GraphicComponent L1;
               mech.GetParentLink(0, out L1);
               Part attacher = (Part)L1;
               attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));
            }
            else
            {
               string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU5000Kg Risers\\";
               mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU5000_" +
Convert.ToString(Convert.ToInt32(settings.skyhook_throw) + 0) + "H.rslib", true);
               Part Riser = (Part)mylib.RootComponent.CopyInstance();
               Riser.Name = "Riser";
               station.GraphicComponents.Add(Riser);
               Riser.Transform.RY = -90 * (Math.PI / 180);
               Riser.Transform.RZ = -180 * (Math.PI / 180);
               Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
               Riser.Transform.X = (-1811.439 - (Convert.ToInt32(settings.skyhook_throw) - 1500)) * .001;

               GraphicComponent L1;
               mech.GetParentLink(0, out L1);
               Part attacher = (Part)L1;
               attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));
            }
         }
```

```
        if (settings.skyhook_capacity == "3000Kg")
        {
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;
            System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\SkyHook\\3000Kg\\");

            //Load Base
            string Path_base;
            string name_base = "3000Kg_SH_Base";
            Path_base = file_base.ToString() + "3000Kg_SH_Base.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base;
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 1500) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 500) * .001;

            //Load Arm
            String Path_arm;
            String name_arm = "3000Kg_SH_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
            Path_arm = file_base.ToString();
            Path_arm = Path_arm + "3000Kg_SH_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
            GraphicComponent Arm = null;
            if (File.Exists(Path_arm))
            {

                mylib = GraphicComponentLibrary.Load(Path_arm, true);
                station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
                station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
                Arm.DisconnectFromLibrary();
            }
            else
            {
                Logger.AddMessage(new LogMessage("This is not a valid selection"));
                Logger.AddMessage(new LogMessage("For a 3000Kg capacity the options are as follows"));
                Logger.AddMessage(new LogMessage("Throw: 1500, 1600, 1700, 1800, 1900, 20000"));
                Logger.AddMessage(new LogMessage("Drop: 500, 600, 700, 800, 900, 1000"));
                return;
            }

            //Load Platter
            String Path_Platter;
```

```
            String name_Platter = "3000Kg_SH_Platter";
            Path_Platter = file_base.Append("3000Kg_SH_Platter.rslib").ToString();
            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "SkyHook";
            mb.AddLink("Base", Base);
            mb.AddLink("Link1", Arm);
            mb.AddLink("Link2", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Link1", new Vector3(1, 0, (Convert.ToInt32(settings.skyhook_drop)
* .001)), new Vector3(0, 0, (Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational,
true);
            mb.AddJoint("J2", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
            mb.SetJointLimit("J1", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
            mb.SetJointLimit("J2", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "SkyHook";

            //Load Riser
            //Load Riser
            if ((Convert.ToInt32(settings.skyhook_drop) + 800) > Convert.ToInt32(settings.skyhook_throw))
            {
               string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU10000Kg Risers\\";
               mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU10000_" +
Convert.ToString(Convert.ToInt32(settings.skyhook_drop) + 800) + "H.rslib", true);
               Part Riser = (Part)mylib.RootComponent.CopyInstance();
               Riser.Name = "Riser";
               station.GraphicComponents.Add(Riser);
               Riser.Transform.RY = -90 * (Math.PI / 180);
               Riser.Transform.RZ = -180 * (Math.PI / 180);
               Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) + 29.59) * .001;
               Riser.Transform.X = (-1916.819 - (Convert.ToInt32(settings.skyhook_throw) - 1500)) * .001;

               GraphicComponent L1;
               mech.GetParentLink(0, out L1);
               Part attacher = (Part)L1;
               attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));
            }
            else
            {
```

```
            string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU10000Kg Risers\\";
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU10000_" +
Convert.ToString(Convert.ToInt32(settings.skyhook_throw) + 0) + "H.rslib", true);
            Part Riser = (Part)mylib.RootComponent.CopyInstance();
            Riser.Name = "Riser";
            station.GraphicComponents.Add(Riser);
            Riser.Transform.RY = -90 * (Math.PI / 180);
            Riser.Transform.RZ = -180 * (Math.PI / 180);
            Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) + 29.59) * .001;
            Riser.Transform.X = (-1916.819 - (Convert.ToInt32(settings.skyhook_throw) - 1500)) * .001;

            GraphicComponent L1;
            mech.GetParentLink(0, out L1);
            Part attacher = (Part)L1;
            attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));
        }
    }

    if (settings.skyhook_capacity == "5000Kg")
    {
        Station station = Project.ActiveProject as Station;
        GraphicComponentLibrary mylib;
        System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\SkyHook\\5000Kg\\");

        //Load Base
        string Path_base;
        string name_base = "5000Kg_SH_Base";
        Path_base = file_base.ToString() + "5000Kg_SH_Base.rslib";
        mylib = GraphicComponentLibrary.Load(Path_base, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        GraphicComponent Base;
        station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
        Base.DisconnectFromLibrary();
        Base.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 1500) * .001;
        Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 500) * .001;

        //Load Arm
        String Path_arm;
        String name_arm = "5000Kg_SH_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
        Path_arm = file_base.ToString();
        Path_arm = Path_arm + "5000Kg_SH_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
        GraphicComponent Arm = null;
        if (File.Exists(Path_arm))
        {
```

```
            mylib = GraphicComponentLibrary.Load(Path_arm, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
            Arm.DisconnectFromLibrary();
        }
        else
        {
            Logger.AddMessage(new LogMessage("This is not a valid selection"));
            Logger.AddMessage(new LogMessage("For a 5000Kg capacity the options are as follows"));
            Logger.AddMessage(new LogMessage("Throw: 1500, 1600, 1700, 1800, 1900, 20000"));
            Logger.AddMessage(new LogMessage("Drop: 500, 600, 700, 800, 900, 1000"));
            return;
        }

        //Load Platter
        String Path_Platter;
        String name_Platter = "5000Kg_SH_Platter";
        Path_Platter = file_base.Append("5000Kg_SH_Platter.rslib").ToString();
        mylib = GraphicComponentLibrary.Load(Path_Platter, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        GraphicComponent Platter;
        station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
        Platter.DisconnectFromLibrary();

        MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
        mb.ModelName = "SkyHook";
        mb.AddLink("Base", Base);
        mb.AddLink("Link1", Arm);
        mb.AddLink("Link2", Platter);
        mb.BaseLink = "Base";
        mb.AddJoint("J1", "Base", "Link1", new Vector3(1, 0, (Convert.ToInt32(settings.skyhook_drop)
* .001)), new Vector3(0, 0, (Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational,
true);
        mb.AddJoint("J2", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
        mb.SetJointLimit("J1", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
        mb.SetJointLimit("J2", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
        Mechanism mech = mb.CompileMechanism();
        station.GraphicComponents.Add(mech);
        mech.Name = "SkyHook";

        //Load Riser
        if ((Convert.ToInt32(settings.skyhook_drop) + 800) > Convert.ToInt32(settings.skyhook_throw))
        {
            string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU10000Kg Risers\\";
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU10000_" +
Convert.ToString(Convert.ToInt32(settings.skyhook_drop) + 800) + "H.rslib", true);
```

```
            Part Riser = (Part)mylib.RootComponent.CopyInstance();
            Riser.Name = "Riser";
            station.GraphicComponents.Add(Riser);
            Riser.Transform.RY = -90 * (Math.PI / 180);
            Riser.Transform.RZ = -180 * (Math.PI / 180);
            Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) + 29.59) * .001;
            Riser.Transform.X = (-1916.819 - (Convert.ToInt32(settings.skyhook_throw) - 1500)) * .001;

            GraphicComponent L1;
            mech.GetParentLink(0, out L1);
            Part attacher = (Part)L1;
            attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));
          }
          else
          {
            string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU10000Kg Risers\\";
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU10000_" +
Convert.ToString(Convert.ToInt32(settings.skyhook_throw) + 0) + "H.rslib", true);
            Part Riser = (Part)mylib.RootComponent.CopyInstance();
            Riser.Name = "Riser";
            station.GraphicComponents.Add(Riser);
            Riser.Transform.RY = -90 * (Math.PI / 180);
            Riser.Transform.RZ = -180 * (Math.PI / 180);
            Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) + 29.59) * .001;
            Riser.Transform.X = (-1916.819 - (Convert.ToInt32(settings.skyhook_throw) - 1500)) * .001;

            GraphicComponent L1;
            mech.GetParentLink(0, out L1);
            Part attacher = (Part)L1;
            attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));
          }
        }

        if (settings.skyhook_capacity == "10000Kg")
        {
          Station station = Project.ActiveProject as Station;
          GraphicComponentLibrary mylib;
          System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\SkyHook\\10000Kg\\");

          //Load Base
          string Path_base;
          string name_base = "10000Kg_SH_Base";
          Path_base = file_base.ToString() + "10000Kg_SH_Base.rslib";
          mylib = GraphicComponentLibrary.Load(Path_base, true);
          station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
          GraphicComponent Base;
```

```
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 500) * .001;
            if (settings.skyhook_drop == "1000" & settings.skyhook_throw != "1500")
            {
               Base.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 1609.53) * .001;
            }
            else
            {
               Base.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 1500) * .001;
            }



            //Load Arm
            String Path_arm;
            String name_arm = "10000Kg_SH_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
            Path_arm = file_base.ToString();
            Path_arm = Path_arm + "10000Kg_SH_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
            GraphicComponent Arm = null;
            if (File.Exists(Path_arm))
            {
               mylib = GraphicComponentLibrary.Load(Path_arm, true);
               station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
               station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
               Arm.DisconnectFromLibrary();
            }
            else
            {
               Logger.AddMessage(new LogMessage("This is not a valid selection"));
               Logger.AddMessage(new LogMessage("For a 10000Kg capacity the options are as follows"));
               Logger.AddMessage(new LogMessage("Throw: 1500, 1600, 1700, 1800, 1900, 2000"));
               Logger.AddMessage(new LogMessage("Drop: 500, 600, 700, 800, 900, 1000"));
               Logger.AddMessage(new LogMessage("If Drop 1000 -> Throw: 1500, 1600, 1700, 1800, 1900,
2000, 2100, 2200, 2300, 2400, 2500, 2600, 2700, 2800, 2900, 3000"));

               GraphicComponent temp;
               station.GraphicComponents.TryGetGraphicComponent(name_base, out temp);
               if (temp != null)
               {
                  station.GraphicComponents.Remove(temp);
               }
               return;
            }

            //Load Platter
```

```
            String Path_Platter;
            String name_Platter = "10000Kg_SH_Platter";
            Path_Platter = file_base.Append("10000Kg_SH_Platter.rslib").ToString();
            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "SkyHook";
            mb.AddLink("Base", Base);
            mb.AddLink("Link1", Arm);
            mb.AddLink("Link2", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Link1", new Vector3(1, 0, (Convert.ToInt32(settings.skyhook_drop)
* .001)), new Vector3(0, 0, (Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational,
true);
            mb.AddJoint("J2", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
            mb.SetJointLimit("J1", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
            mb.SetJointLimit("J2", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "SkyHook";

            //Load Riser
            if ((Convert.ToInt32(settings.skyhook_drop) + 800) > Convert.ToInt32(settings.skyhook_throw))
            {
               string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU20000Kg Risers\\";
               if (Convert.ToInt32(settings.skyhook_drop) + 800 < 1800)
               {
                  mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU20000_" + "1800" +
"H.rslib", true);
               }
               else
               {
                  mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU20000_" +
Convert.ToString(Convert.ToInt32(settings.skyhook_drop) + 800) + "H.rslib", true);
               }

               Part Riser = (Part)mylib.RootComponent.CopyInstance();
               Riser.Name = "Riser";
               station.GraphicComponents.Add(Riser);
               Riser.Transform.RY = -90 * (Math.PI / 180);
               Riser.Transform.RZ = -180 * (Math.PI / 180);
               Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) + 33.93) * .001;
```

```
            Riser.Transform.X = (-(Convert.ToInt32(settings.skyhook_throw)) - 588.32) * .001;

            GraphicComponent L1;
            mech.GetParentLink(0, out L1);
            Part attacher = (Part)L1;
            attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));
          }
          else
          {
            string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU20000Kg Risers\\";
            if (Convert.ToInt32(settings.skyhook_throw) < 1800)
            {
              mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU20000_" + "1800" +
"H.rslib", true);
            }
            else
            {
              mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU20000_" +
Convert.ToString(Convert.ToInt32(settings.skyhook_throw) + 0) + "H.rslib", true);
            }
            Part Riser = (Part)mylib.RootComponent.CopyInstance();
            Riser.Name = "Riser";
            station.GraphicComponents.Add(Riser);
            Riser.Transform.RY = -90 * (Math.PI / 180);
            Riser.Transform.RZ = -180 * (Math.PI / 180);
            Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) + 33.93) * .001;
            Riser.Transform.X = (-(Convert.ToInt32(settings.skyhook_throw)) - 588.32) * .001;

            GraphicComponent L1;
            mech.GetParentLink(0, out L1);
            Part attacher = (Part)L1;
            attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));
          }
        }
      }

      public static void Build2()
      {
          Station station = Project.ActiveProject as Station;
          GraphicComponentLibrary mylib;

          //Load HS
          System.Text.StringBuilder file_HS = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\WTU Headstocks\\Wolf_");
          string Path_HS;
          string name_HS = "Wolf_" + settings.HSTS_capacity.ToString() + "_HS";
          Path_HS = file_HS.ToString() + settings.HSTS_capacity.ToString() + "_HS.rslib";
```

```
            mylib = GraphicComponentLibrary.Load(Path_HS, true);
            Mechanism HS = (Mechanism)mylib.RootComponent.CopyInstance();
            station.GraphicComponents.Add(HS);
            HS.DisconnectFromLibrary();
            HS.Name = "Head Stock";

            //Load HS Riser
            System.Text.StringBuilder file_HS_Riser = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\Riser\\WTU");
            string Path_HS_Riser;
            string temp = settings.HSTS_capacity.ToString();
            temp = temp.TrimEnd('g');
            temp = temp.TrimEnd('K');
            string name_HS_Riser = "Riser_WTU" + temp + "_" + settings.HSTS_riser_height + "H";
            Path_HS_Riser = file_HS_Riser.ToString() + settings.HSTS_capacity + " Risers\\" +
name_HS_Riser + ".rslib";
            mylib = GraphicComponentLibrary.Load(Path_HS_Riser, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent HS_Riser;
            station.GraphicComponents.TryGetGraphicComponent(name_HS_Riser, out HS_Riser);
            HS_Riser.DisconnectFromLibrary();
            HS_Riser.Transform.RY = (-90 * (Math.PI / 180));
            HS_Riser.Transform.RZ = (180 * (Math.PI / 180));
            HS_Riser.Name = "Head Stock Riser";

            if (settings.HSTS_capacity == "3000Kg" | settings.HSTS_capacity == "5000Kg")
            {
               HS_Riser.Transform.Z = (569.91) * .001;
            }
            if (settings.HSTS_capacity == "10000Kg")
            {
               HS_Riser.Transform.Z = (647.48) * .001;
            }
            if (settings.HSTS_capacity == "20000Kg")
            {
               HS_Riser.Transform.Z = (1105.97) * .001;
            }
            if (settings.HSTS_capacity == "30000Kg")
            {
               HS_Riser.Transform.Z = (1197.87) * .001;
            }
            if (settings.HSTS_capacity == "50000Kg")
            {
               HS_Riser.Transform.RY = 0;
               HS_Riser.Transform.RZ = (90 * (Math.PI / 180));
               HS_Riser.Transform.X = (-1093.50) * .001;
            }
            if (settings.HSTS_capacity == "70000Kg")
```

```
        {
            HS_Riser.Transform.Z = (1514.04) * .001;
        }

        //Attach Riser to HS
        GraphicComponent L1;
        HS.GetParentLink(0, out L1);
        Part attacher = (Part)L1;
        attacher.Attach(HS_Riser, false, new Matrix4(new Vector3(0, 0, 0)));

        //Load TS
        if (settings.HSTS_capacity == "3000Kg" | settings.HSTS_capacity == "5000Kg")
        {
            System.Text.StringBuilder file_TS = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\WTU
Tailstocks\\Wolf_3000,5000TS\\WOLF_3000-5000TS_Solid_");
            string Path_TS;
            string name_TS = "TS-" + settings.HSTS_riser_height;
            Path_TS = file_TS.ToString() + settings.HSTS_riser_height.ToString() + ".rslib";
            mylib = GraphicComponentLibrary.Load(Path_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent TS;
            station.GraphicComponents.TryGetGraphicComponent(name_TS, out TS);
            TS.DisconnectFromLibrary();
            TS.Name = "Tail Stock";
        }
        else
        {
            System.Text.StringBuilder file_TS = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\WTU Tailstocks\\");
            string Path_TS;
            string temp2 = settings.HSTS_capacity.ToString();
            temp2 = temp2.TrimEnd('g');
            temp2 = temp2.TrimEnd('K');
            string name_TS = "Wolf_" + temp2 + "TS";
            Path_TS = file_TS.ToString() + name_TS + ".rslib";
            mylib = GraphicComponentLibrary.Load(Path_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent TS;
            station.GraphicComponents.TryGetGraphicComponent(name_TS, out TS);
            TS.DisconnectFromLibrary();
            TS.Name = "Tail Stock";
        }

        //Load TS Riser if needed
        if (settings.HSTS_capacity == "3000Kg" | settings.HSTS_capacity == "5000Kg")
        {
        }
```

```
            else
            {
                System.Text.StringBuilder file_TS_Riser = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\Riser\\WTU");
                string Path_TS_Riser;
                string temp3 = settings.HSTS_capacity.ToString();
                temp3 = temp3.TrimEnd('g');
                temp3 = temp3.TrimEnd('K');
                string name_TS_Riser = "Riser_WTU" + temp3 + "_" + settings.HSTS_riser_height + "H";
                Path_TS_Riser = file_TS_Riser.ToString() + settings.HSTS_capacity + " Risers\\" +
name_TS_Riser + ".rslib";
                mylib = GraphicComponentLibrary.Load(Path_HS_Riser, true);
                station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
                GraphicComponent TS_Riser;
                station.GraphicComponents.TryGetGraphicComponent(name_TS_Riser, out TS_Riser);
                TS_Riser.DisconnectFromLibrary();
                TS_Riser.Transform.RY = (-90 * (Math.PI / 180));
                TS_Riser.Name = "Tail Stock Riser";

                if (settings.HSTS_capacity == "10000Kg")
                {
                    TS_Riser.Transform.Z = (647.48) * .001;
                }
                if (settings.HSTS_capacity == "20000Kg")
                {
                    TS_Riser.Transform.Z = (1105.97) * .001;
                }
                if (settings.HSTS_capacity == "30000Kg")
                {
                    TS_Riser.Transform.Z = (1197.87) * .001;
                }
                if (settings.HSTS_capacity == "50000Kg")
                {
                    TS_Riser.Transform.RY = 0;
                    TS_Riser.Transform.RZ = (90 * (Math.PI / 180));
                    TS_Riser.Transform.X = (+1093.50) * .001;
                }
                if (settings.HSTS_capacity == "70000Kg")
                {
                    TS_Riser.Transform.Z = (1514.04) * .001;
                }

                //Name Tail Stock
                TS_Riser.Name = "Tail Stock Riser";

                //Attach Riser to HS
                GraphicComponent L2;
                station.GraphicComponents.TryGetGraphicComponent("Tail Stock", out L2);
```

```
            Part attacher2 = (Part)L2;
            attacher2.Attach(TS_Riser, false, new Matrix4(new Vector3(0, 0, 0)));
        }

    }

    public static void Build3()
    {
        if (settings.skyhook_capacity == "1000Kg")
        {
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;

            System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\SkyHook\\1000Kg\\");

            //Load Base
            string Path_base;
            string name_base = "1000Kg_SH_Base_Lift";
            Path_base = file_base.ToString() + "1000Kg_SH_Base_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base;
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Platter with Lift
            String Path_Platter_Lift;
            String name_Platter_Lift = "1000Kg_SH_Platter_Lift";
            Path_Platter_Lift = file_base.ToString() + "1000Kg_SH_Platter_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter_Lift, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter_Lift;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter_Lift, out Platter_Lift);
            Platter_Lift.DisconnectFromLibrary();
            Platter_Lift.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Platter_Lift.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;


            //Load Arm
            String Path_arm;
            String name_arm = "1000Kg_SH_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
            Path_arm = file_base.ToString();
            Path_arm = Path_arm + "1000Kg_SH_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
```

```
            GraphicComponent Arm = null;

            if (File.Exists(Path_arm))
            {
               mylib = GraphicComponentLibrary.Load(Path_arm, true);
               station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
               station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
               Arm.DisconnectFromLibrary();
            }

            //Load Platter
            String Path_Platter;
            String name_Platter = "1000Kg_SH_Platter";
            Path_Platter = file_base + "1000Kg_SH_Platter.rslib";

            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "SkyHook";
            mb.AddLink("Base", Base);
            mb.AddLink("Platter_Lift", Platter_Lift);
            mb.AddLink("Link1", Arm);
            mb.AddLink("Link2", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Platter_Lift", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Prismatic, true);
            mb.AddJoint("J2", "Platter_Lift", "Link1", new Vector3(1, 0,
(Convert.ToInt32(settings.skyhook_drop) * .001)), new Vector3(0, 0,
(Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational, true);
            mb.AddJoint("J3", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
            mb.SetJointLimit("J1", 0, 1425 * .001);
            mb.SetJointLimit("J2", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
            mb.SetJointLimit("J3", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "SkyHook";

         }


         if (settings.skyhook_capacity == "3000Kg")
         {
            Station station = Project.ActiveProject as Station;
```

```
        GraphicComponentLibrary mylib;
        System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\SkyHook\\3000Kg\\");

        //Load Base
        string Path_base;
        string name_base = "3000Kg_SH_Base_Lift";
        Path_base = file_base.ToString() + "3000Kg_SH_Base_Lift.rslib";
        mylib = GraphicComponentLibrary.Load(Path_base, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        GraphicComponent Base;
        station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
        Base.DisconnectFromLibrary();
        Base.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 1500) * .001;
        Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 500) * .001;

        //Load Platter with Lift
        String Path_Platter_Lift;
        String name_Platter_Lift = "3000Kg_SH_Platter_Lift";
        Path_Platter_Lift = file_base.ToString() + "3000Kg_SH_Platter_Lift.rslib";
        mylib = GraphicComponentLibrary.Load(Path_Platter_Lift, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        GraphicComponent Platter_Lift;
        station.GraphicComponents.TryGetGraphicComponent(name_Platter_Lift, out Platter_Lift);
        Platter_Lift.DisconnectFromLibrary();
        Platter_Lift.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 1500) * .001;
        Platter_Lift.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 500) * .001;

        //Load Arm
        String Path_arm;
        String name_arm = "3000Kg_SH_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
        Path_arm = file_base.ToString();
        Path_arm = Path_arm + "3000Kg_SH_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
        GraphicComponent Arm = null;
        if (File.Exists(Path_arm))
        {

          mylib = GraphicComponentLibrary.Load(Path_arm, true);
          station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
          station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
          Arm.DisconnectFromLibrary();
        }

        //Load Platter
        String Path_Platter;
        String name_Platter = "3000Kg_SH_Platter";
```

```
            Path_Platter = file_base.Append("3000Kg_SH_Platter.rslib").ToString();
            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "SkyHook";
            mb.AddLink("Base", Base);
            mb.AddLink("Platter_Lift", Platter_Lift);
            mb.AddLink("Link1", Arm);
            mb.AddLink("Link2", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Platter_Lift", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Prismatic, true);
            mb.AddJoint("J2", "Platter_Lift", "Link1", new Vector3(1, 0,
(Convert.ToInt32(settings.skyhook_drop) * .001)), new Vector3(0, 0,
(Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational, true);
            mb.AddJoint("J3", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
            mb.SetJointLimit("J1", 0, 1425 * .001);
            mb.SetJointLimit("J2", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
            mb.SetJointLimit("J3", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "SkyHook";

        }

        if (settings.skyhook_capacity == "5000Kg")
        {
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;
            System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\SkyHook\\5000Kg\\");

            //Load Base
            string Path_base;
            string name_base = "5000Kg_SH_Base_Lift";
            Path_base = file_base.ToString() + "5000Kg_SH_Base_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base;
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 1500) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 500) * .001;
```

```
            //Load Platter with Lift
            String Path_Platter_Lift;
            String name_Platter_Lift = "5000Kg_SH_Platter_Lift";
            Path_Platter_Lift = file_base.ToString() + "5000Kg_SH_Platter_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter_Lift, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter_Lift;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter_Lift, out Platter_Lift);
            Platter_Lift.DisconnectFromLibrary();
            Platter_Lift.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 1500) * .001;
            Platter_Lift.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 500) * .001;

            //Load Arm
            String Path_arm;
            String name_arm = "5000Kg_SH_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
            Path_arm = file_base.ToString();
            Path_arm = Path_arm + "5000Kg_SH_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
            GraphicComponent Arm = null;
            if (File.Exists(Path_arm))
            {
               mylib = GraphicComponentLibrary.Load(Path_arm, true);
               station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
               station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
               Arm.DisconnectFromLibrary();
            }

            //Load Platter
            String Path_Platter;
            String name_Platter = "5000Kg_SH_Platter";
            Path_Platter = file_base.Append("5000Kg_SH_Platter.rslib").ToString();
            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "SkyHook";
            mb.AddLink("Base", Base);
            mb.AddLink("Platter_Lift", Platter_Lift);
            mb.AddLink("Link1", Arm);
            mb.AddLink("Link2", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Platter_Lift", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Prismatic, true);
```

```
            mb.AddJoint("J2", "Platter_Lift", "Link1", new Vector3(1, 0,
(Convert.ToInt32(settings.skyhook_drop) * .001)), new Vector3(0, 0,
(Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational, true);
            mb.AddJoint("J3", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
            mb.SetJointLimit("J1", 0, 1425 * .001);
            mb.SetJointLimit("J2", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
            mb.SetJointLimit("J3", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "SkyHook";
        }
    }

    public static void Build4()
    {
        if (settings.skyhook_capacity == "5000Kg")
        {
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;

            System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\Drop_Center\\5000Kg\\");

            //Load Base
            string Path_base;
            string name_base = "5000Kg_DC_Base";
            Path_base = file_base.ToString() + "5000Kg_DC_Base.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base;
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.X = -(.5 * Convert.ToInt32(settings.skyhook_throw) + 302.25) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;

            //Load Arm
            String Path_arm;
            String name_arm = "5000Kg_DC_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
            Path_arm = file_base.ToString();
            Path_arm = Path_arm + "5000Kg_DC_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
            GraphicComponent Arm = null;

            if (File.Exists(Path_arm))
            {
                mylib = GraphicComponentLibrary.Load(Path_arm, true);
```

```
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
            Arm.DisconnectFromLibrary();
        }

        //Load Platter
        String Path_Platter;
        String name_Platter = "5000Kg_DC_Platter";
        Path_Platter = file_base + "5000Kg_DC_Platter.rslib";

        mylib = GraphicComponentLibrary.Load(Path_Platter, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        GraphicComponent Platter;
        station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
        Platter.DisconnectFromLibrary();

        MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
        mb.ModelName = "DropCenter";
        mb.AddLink("Base", Base);
        mb.AddLink("Link1", Arm);
        mb.AddLink("Link2", Platter);
        mb.BaseLink = "Base";
        mb.AddJoint("J1", "Base", "Link1", new Vector3(1, 0, (Convert.ToInt32(settings.skyhook_drop)
* .001)), new Vector3(0, 0, (Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational,
true);
        mb.AddJoint("J2", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
        mb.SetJointLimit("J1", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
        mb.SetJointLimit("J2", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
        Mechanism mech = mb.CompileMechanism();
        station.GraphicComponents.Add(mech);
        mech.Name = "DropCenter";

        //Load Base 2
        mylib = GraphicComponentLibrary.Load(Path_base, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
        Base.DisconnectFromLibrary();
        Base.Transform.RZ = 0;
        Base.Transform.X = +(.5 * Convert.ToInt32(settings.skyhook_throw) + 302.25) * .001;
        Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
        GraphicComponent L1;
        mech.GetParentLink(0, out L1);
        Part attacher = (Part)L1;
        attacher.Attach(Base, false, new Matrix4(new Vector3(0, 0, 0)));

        //Load Riser
```

```
            string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU5000Kg Risers\\";
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU5000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser = (Part)mylib.RootComponent.CopyInstance();
            Riser.Name = "Riser";
            station.GraphicComponents.Add(Riser);
            Riser.Transform.RY = -90 * (Math.PI / 180);
            Riser.Transform.RZ = 0;
            Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            Riser.Transform.X = ((.5 * Convert.ToInt32(settings.skyhook_throw) + 302.25)) * .001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));

            //Load Riser2
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU5000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser2 = (Part)mylib.RootComponent.CopyInstance();
            Riser2.Name = "Riser";
            station.GraphicComponents.Add(Riser2);
            Riser2.Transform.RY = -90 * (Math.PI / 180);
            Riser2.Transform.RZ = -180 * (Math.PI / 180);
            Riser2.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            Riser2.Transform.X = -((.5 * Convert.ToInt32(settings.skyhook_throw) + 302.25)) * .001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser2, false, new Matrix4(new Vector3(0, 0, 0)));
        }

        if (settings.skyhook_capacity == "10000Kg")
        {
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;

            System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Drop_Center\\10000Kg\\");

            //Load Base
            string Path_base;
            string name_base = "10000Kg_DC_Base";
            Path_base = file_base.ToString() + "10000Kg_DC_Base.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base;
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.X = -(.5 * Convert.ToInt32(settings.skyhook_throw) + 302.25) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
```

```
            //Load Arm
            String Path_arm;
            String name_arm = "10000Kg_DC_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
            Path_arm = file_base.ToString();
            Path_arm = Path_arm + "10000Kg_DC_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
            GraphicComponent Arm = null;

            if (File.Exists(Path_arm))
            {
               mylib = GraphicComponentLibrary.Load(Path_arm, true);
               station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
               station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
               Arm.DisconnectFromLibrary();
            }

            //Load Platter
            String Path_Platter;
            String name_Platter = "10000Kg_DC_Platter";
            Path_Platter = file_base + "10000Kg_DC_Platter.rslib";

            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "DropCenter";
            mb.AddLink("Base", Base);
            mb.AddLink("Link1", Arm);
            mb.AddLink("Link2", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Link1", new Vector3(1, 0, (Convert.ToInt32(settings.skyhook_drop)
* .001)), new Vector3(0, 0, (Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational,
true);
            mb.AddJoint("J2", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
            mb.SetJointLimit("J1", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
            mb.SetJointLimit("J2", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "DropCenter";

            //Load Base 2
            mylib = GraphicComponentLibrary.Load(Path_base, true);
```

```
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.RZ = 0;
            Base.Transform.X = +(.5 * Convert.ToInt32(settings.skyhook_throw) + 302.25) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            GraphicComponent L1;
            mech.GetParentLink(0, out L1);
            Part attacher = (Part)L1;
            attacher.Attach(Base, false, new Matrix4(new Vector3(0, 0, 0)));

            //Load Riser
            string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU10000Kg Risers\\";
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU10000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser = (Part)mylib.RootComponent.CopyInstance();
            Riser.Name = "Riser";
            station.GraphicComponents.Add(Riser);
            Riser.Transform.RY = -90 * (Math.PI / 180);
            Riser.Transform.RZ = 0;
            Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            Riser.Transform.X = (.5 * (Convert.ToInt32(settings.skyhook_throw) + 540.36)) * .001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));

            //Load Riser2
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU10000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser2 = (Part)mylib.RootComponent.CopyInstance();
            Riser2.Name = "Riser";
            station.GraphicComponents.Add(Riser2);
            Riser2.Transform.RY = -90 * (Math.PI / 180);
            Riser2.Transform.RZ = -180 * (Math.PI / 180);
            Riser2.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            Riser2.Transform.X = -(.5 * (Convert.ToInt32(settings.skyhook_throw) + 540.36)) * .001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser2, false, new Matrix4(new Vector3(0, 0, 0)));

        }

        if (settings.skyhook_capacity == "20000Kg")
        {
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;
```

```
        System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Drop_Center\\20000Kg\\");

        //Load Base
        string Path_base;
        string name_base = "20000Kg_DC_Base";
        Path_base = file_base.ToString() + "20000Kg_DC_Base.rslib";
        mylib = GraphicComponentLibrary.Load(Path_base, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        GraphicComponent Base;
        station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
        Base.DisconnectFromLibrary();
        Base.Transform.X = -(.5 * Convert.ToInt32(settings.skyhook_throw) + 302.25) * .001;
        Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;

        //Load Arm
        String Path_arm;
        String name_arm = "20000Kg_DC_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
        Path_arm = file_base.ToString();
        Path_arm = Path_arm + "20000Kg_DC_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
        GraphicComponent Arm = null;

        if (File.Exists(Path_arm))
        {
           mylib = GraphicComponentLibrary.Load(Path_arm, true);
           station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
           station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
           Arm.DisconnectFromLibrary();
        }

        //Load Platter
        String Path_Platter;
        String name_Platter = "20000Kg_DC_Platter";
        Path_Platter = file_base + "20000Kg_DC_Platter.rslib";

        mylib = GraphicComponentLibrary.Load(Path_Platter, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        GraphicComponent Platter;
        station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
        Platter.DisconnectFromLibrary();

        MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
        mb.ModelName = "DropCenter";
        mb.AddLink("Base", Base);
        mb.AddLink("Link1", Arm);
```

```
            mb.AddLink("Link2", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Link1", new Vector3(1, 0, (Convert.ToInt32(settings.skyhook_drop)
* .001)), new Vector3(0, 0, (Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational,
true);
            mb.AddJoint("J2", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
            mb.SetJointLimit("J1", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
            mb.SetJointLimit("J2", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "DropCenter";

            //Load Base 2
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.RZ = 0;
            Base.Transform.X = +(.5 * Convert.ToInt32(settings.skyhook_throw) + 302.25) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            GraphicComponent L1;
            mech.GetParentLink(0, out L1);
            Part attacher = (Part)L1;
            attacher.Attach(Base, false, new Matrix4(new Vector3(0, 0, 0)));

            //Load Riser
            string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU20000Kg Risers\\";
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU20000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser = (Part)mylib.RootComponent.CopyInstance();
            Riser.Name = "Riser";
            station.GraphicComponents.Add(Riser);
            Riser.Transform.RY = -90 * (Math.PI / 180);
            Riser.Transform.RZ = -180 * (Math.PI / 180);
            Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            Riser.Transform.X = (.5 * (Convert.ToInt32(settings.skyhook_throw) + 3885.09)) * .001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));

            //Load Riser2
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU20000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser2 = (Part)mylib.RootComponent.CopyInstance();
            Riser2.Name = "Riser";
            station.GraphicComponents.Add(Riser2);
            Riser2.Transform.RY = -90 * (Math.PI / 180);
```

```
            Riser2.Transform.RZ = 0;
            Riser2.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            Riser2.Transform.X = -(.5 * (Convert.ToInt32(settings.skyhook_throw) + 3885.09)) * .001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser2, false, new Matrix4(new Vector3(0, 0, 0)));
        }

        if (settings.skyhook_capacity == "30000Kg")
        {
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;

            System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Drop_Center\\30000Kg\\");

            //Load Base
            string Path_base;
            string name_base = "30000Kg_DC_Base";
            Path_base = file_base.ToString() + "30000Kg_DC_Base.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base;
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.X = -(.5 * Convert.ToInt32(settings.skyhook_throw) + 462.92) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;

            //Load Arm
            String Path_arm;
            String name_arm = "30000Kg_DC_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
            Path_arm = file_base.ToString();
            Path_arm = Path_arm + "30000Kg_DC_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
            GraphicComponent Arm = null;

            if (File.Exists(Path_arm))
            {
                mylib = GraphicComponentLibrary.Load(Path_arm, true);
                station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
                station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
                Arm.DisconnectFromLibrary();
            }

            //Load Platter
            String Path_Platter;
            String name_Platter = "30000Kg_DC_Platter";
```

```
            Path_Platter = file_base + "30000Kg_DC_Platter.rslib";

            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "DropCenter";
            mb.AddLink("Base", Base);
            mb.AddLink("Link1", Arm);
            mb.AddLink("Link2", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Link1", new Vector3(1, 0, (Convert.ToInt32(settings.skyhook_drop)
* .001)), new Vector3(0, 0, (Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational,
true);
            mb.AddJoint("J2", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
            mb.SetJointLimit("J1", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
            mb.SetJointLimit("J2", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "DropCenter";

            //Load Base 2
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.RZ = 0;
            Base.Transform.X = +(.5 * Convert.ToInt32(settings.skyhook_throw) + 462.92) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            GraphicComponent L1;
            mech.GetParentLink(0, out L1);
            Part attacher = (Part)L1;
            attacher.Attach(Base, false, new Matrix4(new Vector3(0, 0, 0)));

            //Load Riser
            string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU30000Kg Risers\\";
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU30000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser = (Part)mylib.RootComponent.CopyInstance();
            Riser.Name = "Riser";
            station.GraphicComponents.Add(Riser);
            Riser.Transform.RY = -90 * (Math.PI / 180);
            Riser.Transform.RZ = -180 * (Math.PI / 180);
```

```
            Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            Riser.Transform.X = (.5 * (Convert.ToInt32(settings.skyhook_throw) + 4638.35)) * .001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));

            //Load Riser2
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU30000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser2 = (Part)mylib.RootComponent.CopyInstance();
            Riser2.Name = "Riser";
            station.GraphicComponents.Add(Riser2);
            Riser2.Transform.RY = -90 * (Math.PI / 180);
            Riser2.Transform.RZ = 0;
            Riser2.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            Riser2.Transform.X = -(.5 * (Convert.ToInt32(settings.skyhook_throw) + 4638.35)) * .001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser2, false, new Matrix4(new Vector3(0, 0, 0)));
        }

        if (settings.skyhook_capacity == "50000Kg")
        {
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;

            System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Drop_Center\\50000Kg\\");

            //Load Base
            string Path_base;
            string name_base = "50000Kg_DC_Base";
            Path_base = file_base.ToString() + "50000Kg_DC_Base.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base;
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.X = -(.5 * Convert.ToInt32(settings.skyhook_throw) + 514.57) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;

            //Load Arm
            String Path_arm;
            String name_arm = "50000Kg_DC_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
            Path_arm = file_base.ToString();
            Path_arm = Path_arm + "50000Kg_DC_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
            GraphicComponent Arm = null;
```

```
            if (File.Exists(Path_arm))
            {
               mylib = GraphicComponentLibrary.Load(Path_arm, true);
               station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
               station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
               Arm.DisconnectFromLibrary();
            }

            //Load Platter
            String Path_Platter;
            String name_Platter = "50000Kg_DC_Platter";
            Path_Platter = file_base + "50000Kg_DC_Platter.rslib";

            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "DropCenter";
            mb.AddLink("Base", Base);
            mb.AddLink("Link1", Arm);
            mb.AddLink("Link2", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Link1", new Vector3(1, 0, (Convert.ToInt32(settings.skyhook_drop)
* .001)), new Vector3(0, 0, (Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational,
true);
            mb.AddJoint("J2", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
            mb.SetJointLimit("J1", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
            mb.SetJointLimit("J2", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "DropCenter";

            //Load Base 2
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.RZ = 0;
            Base.Transform.X = +(.5 * Convert.ToInt32(settings.skyhook_throw) + 514.57) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            GraphicComponent L1;
            mech.GetParentLink(0, out L1);
            Part attacher = (Part)L1;
```

```
            attacher.Attach(Base, false, new Matrix4(new Vector3(0, 0, 0)));

            //Load Riser
            string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU50000Kg Risers\\";
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU50000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser = (Part)mylib.RootComponent.CopyInstance();
            Riser.Name = "Riser";
            station.GraphicComponents.Add(Riser);
            Riser.Transform.RY = 0;
            Riser.Transform.RZ = -90 * (Math.PI / 180);
            Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)-3160.63) * .001;
            Riser.Transform.X = (.5 * (Convert.ToInt32(settings.skyhook_throw) + 3706.65 - 2*245.26)) *
.001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));

            //Load Riser2
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU50000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser2 = (Part)mylib.RootComponent.CopyInstance();
            Riser2.Name = "Riser";
            station.GraphicComponents.Add(Riser2);
            Riser2.Transform.RY = 0;
            Riser2.Transform.RZ = -90 * (Math.PI / 180);
            Riser2.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)-3150.63) * .001;
            Riser2.Transform.X = -(.5 * (Convert.ToInt32(settings.skyhook_throw) + 3706.65 - 2*245.26)) *
.001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser2, false, new Matrix4(new Vector3(0, 0, 0)));
        }

        if (settings.skyhook_capacity == "70000Kg")
        {
            Station station = Project.ActiveProject as Station;
            GraphicComponentLibrary mylib;

            System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Drop_Center\\70000Kg\\");

            //Load Base
            string Path_base;
            string name_base = "70000Kg_DC_Base";
            Path_base = file_base.ToString() + "70000Kg_DC_Base.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
```

```
            GraphicComponent Base;
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.X = -(.5 * Convert.ToInt32(settings.skyhook_throw) + 302.25+676.17/2) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;

            //Load Arm
            String Path_arm;
            String name_arm = "70000Kg_DC_" + settings.skyhook_drop + "D_" + settings.skyhook_throw +
"T_Arm";
            Path_arm = file_base.ToString();
            Path_arm = Path_arm + "70000Kg_DC_" + settings.skyhook_drop + "D_" +
settings.skyhook_throw + "T_Arm.rslib";
            GraphicComponent Arm = null;

            if (File.Exists(Path_arm))
            {
               mylib = GraphicComponentLibrary.Load(Path_arm, true);
               station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
               station.GraphicComponents.TryGetGraphicComponent(name_arm, out Arm);
               Arm.DisconnectFromLibrary();
            }

            //Load Platter
            String Path_Platter;
            String name_Platter = "70000Kg_DC_Platter";
            Path_Platter = file_base + "70000Kg_DC_Platter.rslib";

            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "DropCenter";
            mb.AddLink("Base", Base);
            mb.AddLink("Link1", Arm);
            mb.AddLink("Link2", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Link1", new Vector3(1, 0, (Convert.ToInt32(settings.skyhook_drop)
* .001)), new Vector3(0, 0, (Convert.ToInt32(settings.skyhook_drop) * .001)), JointType.Rotational,
true);
            mb.AddJoint("J2", "Link1", "Link2", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Rotational, true);
            mb.SetJointLimit("J1", -180 * (Math.PI / 180), 180 * (Math.PI / 180));
            mb.SetJointLimit("J2", -360 * (Math.PI / 180), 360 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
```

```
            station.GraphicComponents.Add(mech);
            mech.Name = "DropCenter";

            //Load Base 2
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.RZ = 0;
            Base.Transform.X = +(.5 * Convert.ToInt32(settings.skyhook_throw) + 349.25 - 47 + 676.17/2) *
.001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            GraphicComponent L1;
            mech.GetParentLink(0, out L1);
            Part attacher = (Part)L1;
            attacher.Attach(Base, false, new Matrix4(new Vector3(0, 0, 0)));

            //Load Riser
            string riser_path = "C:\\Program Files (x86)\\Common Files\\ABB Industrial IT\\Robotics
IT\\RobotStudio\\Addins\\Riser\\WTU70000Kg Risers\\";
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU70000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser = (Part)mylib.RootComponent.CopyInstance();
            Riser.Name = "Riser";
            station.GraphicComponents.Add(Riser);
            Riser.Transform.RY = -90 * (Math.PI / 180);
            Riser.Transform.RZ = -180 * (Math.PI / 180);
            Riser.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            Riser.Transform.X = (.5 * (Convert.ToInt32(settings.skyhook_throw) + 6232.11)) * .001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser, false, new Matrix4(new Vector3(0, 0, 0)));

            //Load Riser2
            mylib = GraphicComponentLibrary.Load(riser_path + "Riser_WTU70000_" +
Convert.ToString(settings.DC_riser_height) + "H.rslib", true);
            Part Riser2 = (Part)mylib.RootComponent.CopyInstance();
            Riser2.Name = "Riser";
            station.GraphicComponents.Add(Riser2);
            Riser2.Transform.RY = -90 * (Math.PI / 180);
            Riser2.Transform.RZ = 0;
            Riser2.Transform.Z = (Convert.ToInt32(settings.skyhook_drop)) * .001;
            Riser2.Transform.X = -(.5 * (Convert.ToInt32(settings.skyhook_throw) + 6232.11)) * .001;
            mech.GetParentLink(0, out L1);
            attacher.Attach(Riser2, false, new Matrix4(new Vector3(0, 0, 0)));
        }
    }

    public static void Build5()
```

```
{
    //Initialize
    Station station = Project.ActiveProject as Station;
    GraphicComponentLibrary mylib;



    if (settings.skyhook_capacity == "1000Kg")
    {
        System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\SkyHook\\1000Kg\\");

        //Load Base
        string Path_base;
        string name_base = "1000Kg_SH_Base_Lift";
        Path_base = file_base.ToString() + "1000Kg_SH_Base_Lift.rslib";
        mylib = GraphicComponentLibrary.Load(Path_base, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        GraphicComponent Base;
        station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
        Base.DisconnectFromLibrary();
        Base.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
        Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

        //Load Platter Car with Lift
        String Path_Platter_Lift;
        String name_Platter_Lift = "1000Kg_SH_Platter_Lift";
        Path_Platter_Lift = file_base.ToString() + "1000Kg_SH_Platter_Lift.rslib";
        mylib = GraphicComponentLibrary.Load(Path_Platter_Lift, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        GraphicComponent Platter_Lift;
        station.GraphicComponents.TryGetGraphicComponent(name_Platter_Lift, out Platter_Lift);
        Platter_Lift.DisconnectFromLibrary();
        Platter_Lift.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
        Platter_Lift.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

        //Load Platter
        String Path_Platter;
        String name_Platter = "1000Kg_SH_Platter";
        Path_Platter = file_base + "1000Kg_SH_Platter.rslib";
        mylib = GraphicComponentLibrary.Load(Path_Platter, true);
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        GraphicComponent Platter;
        station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
        Platter.DisconnectFromLibrary();
        Platter.Transform.X = -1848.227*.001;
        Platter.Transform.Z = 499.66*.001;
        Platter.Transform.RY = 90 * (Math.PI / 180);
```

```
            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "HS with Lift";
            mb.AddLink("Base", Base);
            mb.AddLink("Platter_Lift", Platter_Lift);
            mb.AddLink("Platter", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Platter_Lift", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Prismatic, true);
            mb.AddJoint("J2", "Platter_Lift", "Platter", new Vector3(0, 0, 499.66 * .001), new Vector3(1, 0,
499.66 * .001), JointType.Rotational, true);
            mb.SetJointLimit("J1", 0, 1425 * .001);
            mb.SetJointLimit("J2", -720 * (Math.PI / 180), 720 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "HS with Lift";



            //Load TS
            //Load Base
            string Path_base_TS;
            string name_base_TS = "1000Kg_SH_Base_Lift";
            Path_base_TS = file_base.ToString() + "1000Kg_SH_Base_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base_TS;
            station.GraphicComponents.TryGetGraphicComponent(name_base_TS, out Base_TS);
            Base_TS.DisconnectFromLibrary();
            Base_TS.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Base_TS.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Platter Car with Lift
            String Path_Platter_Lift_TS;
            String name_Platter_Lift_TS = "1000Kg_SH_Platter_Lift";
            Path_Platter_Lift_TS = file_base.ToString() + "1000Kg_SH_Platter_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter_Lift_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter_Lift_TS;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter_Lift_TS, out
Platter_Lift_TS);
            Platter_Lift_TS.DisconnectFromLibrary();
            Platter_Lift_TS.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Platter_Lift_TS.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Platter
            String Path_Platter_TS;
            String name_Platter_TS = "1000Kg_SH_Platter";
```

```
            Path_Platter_TS = file_base + "1000Kg_SH_Platter.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter_TS;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter_TS, out Platter_TS);
            Platter_TS.DisconnectFromLibrary();
            Platter_TS.Transform.X = -1848.227 * .001;
            Platter_TS.Transform.Z = 499.66 * .001;
            Platter_TS.Transform.RY = 90 * (Math.PI / 180);

            MechanismBuilder mb_TS = new MechanismBuilder(MechanismType.ExternalAxis);
            mb_TS.ModelName = "TS with Lift";
            mb_TS.AddLink("Base", Base_TS);
            mb_TS.AddLink("Platter_Lift", Platter_Lift_TS);
            mb_TS.AddLink("Platter", Platter_TS);
            mb_TS.BaseLink = "Base";
            mb_TS.AddJoint("J1", "Base", "Platter_Lift", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Prismatic, true);
            mb_TS.AddJoint("J2", "Platter_Lift", "Platter", new Vector3(0, 0, 499.66 * .001), new Vector3(1,
0, 499.66 * .001), JointType.Rotational, true);
            mb_TS.SetJointLimit("J1", 0, 1425 * .001);
            mb_TS.SetJointLimit("J2", -720 * (Math.PI / 180), 720 * (Math.PI / 180));
            Mechanism mech_TS = mb_TS.CompileMechanism();
            station.GraphicComponents.Add(mech_TS);
            mech_TS.Name = "TS with Lift";
            mech_TS.Transform.RZ = 180 * (Math.PI / 180);
        }

        if (settings.skyhook_capacity == "3000Kg")
        {
            System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\SkyHook\\3000Kg\\");

            //Load Base
            string Path_base;
            string name_base = "3000Kg_SH_Base_Lift";
            Path_base = file_base.ToString() + "3000Kg_SH_Base_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base;
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Platter Car with Lift
            String Path_Platter_Lift;
            String name_Platter_Lift = "3000Kg_SH_Platter_Lift";
```

```
            Path_Platter_Lift = file_base.ToString() + "3000Kg_SH_Platter_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter_Lift, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter_Lift;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter_Lift, out Platter_Lift);
            Platter_Lift.DisconnectFromLibrary();
            Platter_Lift.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Platter_Lift.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Platter
            String Path_Platter;
            String name_Platter = "3000Kg_SH_Platter";
            Path_Platter = file_base + "3000Kg_SH_Platter.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();
            Platter.Transform.X = -1848.227 * .001;
            Platter.Transform.Z = 499.66 * .001;
            Platter.Transform.RY = 90 * (Math.PI / 180);

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "HS with Lift";
            mb.AddLink("Base", Base);
            mb.AddLink("Platter_Lift", Platter_Lift);
            mb.AddLink("Platter", Platter);
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Platter_Lift", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Prismatic, true);
            mb.AddJoint("J2", "Platter_Lift", "Platter", new Vector3(0, 0, 499.66 * .001), new Vector3(1, 0,
499.66 * .001), JointType.Rotational, true);
            mb.SetJointLimit("J1", 0, 1425 * .001);
            mb.SetJointLimit("J2", -720 * (Math.PI / 180), 720 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "HS with Lift";



            //Load TS
            //Load Base
            string Path_base_TS;
            string name_base_TS = "3000Kg_SH_Base_Lift";
            Path_base_TS = file_base.ToString() + "3000Kg_SH_Base_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base_TS;
```

```
            station.GraphicComponents.TryGetGraphicComponent(name_base_TS, out Base_TS);
            Base_TS.DisconnectFromLibrary();
            Base_TS.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Base_TS.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Platter Car with Lift
            String Path_Platter_Lift_TS;
            String name_Platter_Lift_TS = "3000Kg_SH_Platter_Lift";
            Path_Platter_Lift_TS = file_base.ToString() + "3000Kg_SH_Platter_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter_Lift_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter_Lift_TS;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter_Lift_TS, out
Platter_Lift_TS);
            Platter_Lift_TS.DisconnectFromLibrary();
            Platter_Lift_TS.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Platter_Lift_TS.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Platter
            String Path_Platter_TS;
            String name_Platter_TS = "3000Kg_SH_Platter";
            Path_Platter_TS = file_base + "3000Kg_SH_Platter.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter_TS;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter_TS, out Platter_TS);
            Platter_TS.DisconnectFromLibrary();
            Platter_TS.Transform.X = -1848.227 * .001;
            Platter_TS.Transform.Z = 499.66 * .001;
            Platter_TS.Transform.RY = 90 * (Math.PI / 180);

            MechanismBuilder mb_TS = new MechanismBuilder(MechanismType.ExternalAxis);
            mb_TS.ModelName = "TS with Lift";
            mb_TS.AddLink("Base", Base_TS);
            mb_TS.AddLink("Platter_Lift", Platter_Lift_TS);
            mb_TS.AddLink("Platter", Platter_TS);
            mb_TS.BaseLink = "Base";
            mb_TS.AddJoint("J1", "Base", "Platter_Lift", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Prismatic, true);
            mb_TS.AddJoint("J2", "Platter_Lift", "Platter", new Vector3(0, 0, 499.66 * .001), new Vector3(1,
0, 499.66 * .001), JointType.Rotational, true);
            mb_TS.SetJointLimit("J1", 0, 1425 * .001);
            mb_TS.SetJointLimit("J2", -720 * (Math.PI / 180), 720 * (Math.PI / 180));
            Mechanism mech_TS = mb_TS.CompileMechanism();
            station.GraphicComponents.Add(mech_TS);
            mech_TS.Name = "TS with Lift";
            mech_TS.Transform.RZ = 180 * (Math.PI / 180);
        }
```

```
        if (settings.skyhook_capacity == "5000Kg")
        {
            System.Text.StringBuilder file_base = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\SkyHook\\5000Kg\\");

            //Load Base
            string Path_base;
            string name_base = "5000Kg_SH_Base_Lift";
            Path_base = file_base.ToString() + "5000Kg_SH_Base_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base;
            station.GraphicComponents.TryGetGraphicComponent(name_base, out Base);
            Base.DisconnectFromLibrary();
            Base.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Base.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Platter Car with Lift
            String Path_Platter_Lift;
            String name_Platter_Lift = "5000Kg_SH_Platter_Lift";
            Path_Platter_Lift = file_base.ToString() + "5000Kg_SH_Platter_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter_Lift, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter_Lift;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter_Lift, out Platter_Lift);
            Platter_Lift.DisconnectFromLibrary();
            Platter_Lift.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Platter_Lift.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Platter
            String Path_Platter;
            String name_Platter = "5000Kg_SH_Platter";
            Path_Platter = file_base + "5000Kg_SH_Platter.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter, out Platter);
            Platter.DisconnectFromLibrary();
            Platter.Transform.X = -1848.227 * .001;
            Platter.Transform.Z = 499.66 * .001;
            Platter.Transform.RY = 90 * (Math.PI / 180);

            MechanismBuilder mb = new MechanismBuilder(MechanismType.ExternalAxis);
            mb.ModelName = "HS with Lift";
            mb.AddLink("Base", Base);
            mb.AddLink("Platter_Lift", Platter_Lift);
            mb.AddLink("Platter", Platter);
```

```
            mb.BaseLink = "Base";
            mb.AddJoint("J1", "Base", "Platter_Lift", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Prismatic, true);
            mb.AddJoint("J2", "Platter_Lift", "Platter", new Vector3(0, 0, 499.66 * .001), new Vector3(1, 0,
499.66 * .001), JointType.Rotational, true);
            mb.SetJointLimit("J1", 0, 1425 * .001);
            mb.SetJointLimit("J2", -720 * (Math.PI / 180), 720 * (Math.PI / 180));
            Mechanism mech = mb.CompileMechanism();
            station.GraphicComponents.Add(mech);
            mech.Name = "HS with Lift";



            //Load TS
            //Load Base
            string Path_base_TS;
            string name_base_TS = "5000Kg_SH_Base_Lift";
            Path_base_TS = file_base.ToString() + "5000Kg_SH_Base_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_base_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Base_TS;
            station.GraphicComponents.TryGetGraphicComponent(name_base_TS, out Base_TS);
            Base_TS.DisconnectFromLibrary();
            Base_TS.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Base_TS.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Platter Car with Lift
            String Path_Platter_Lift_TS;
            String name_Platter_Lift_TS = "5000Kg_SH_Platter_Lift";
            Path_Platter_Lift_TS = file_base.ToString() + "5000Kg_SH_Platter_Lift.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter_Lift_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter_Lift_TS;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter_Lift_TS, out
Platter_Lift_TS);
            Platter_Lift_TS.DisconnectFromLibrary();
            Platter_Lift_TS.Transform.X = -(Convert.ToInt32(settings.skyhook_throw) - 400) * .001;
            Platter_Lift_TS.Transform.Z = (Convert.ToInt32(settings.skyhook_drop) - 400) * .001;

            //Load Platter
            String Path_Platter_TS;
            String name_Platter_TS = "5000Kg_SH_Platter";
            Path_Platter_TS = file_base + "5000Kg_SH_Platter.rslib";
            mylib = GraphicComponentLibrary.Load(Path_Platter_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent Platter_TS;
            station.GraphicComponents.TryGetGraphicComponent(name_Platter_TS, out Platter_TS);
            Platter_TS.DisconnectFromLibrary();
```

```
            Platter_TS.Transform.X = -1848.227 * .001;
            Platter_TS.Transform.Z = 499.66 * .001;
            Platter_TS.Transform.RY = 90 * (Math.PI / 180);

            MechanismBuilder mb_TS = new MechanismBuilder(MechanismType.ExternalAxis);
            mb_TS.ModelName = "TS with Lift";
            mb_TS.AddLink("Base", Base_TS);
            mb_TS.AddLink("Platter_Lift", Platter_Lift_TS);
            mb_TS.AddLink("Platter", Platter_TS);
            mb_TS.BaseLink = "Base";
            mb_TS.AddJoint("J1", "Base", "Platter_Lift", new Vector3(0, 0, 0), new Vector3(0, 0, 1),
JointType.Prismatic, true);
            mb_TS.AddJoint("J2", "Platter_Lift", "Platter", new Vector3(0, 0, 499.66 * .001), new Vector3(1,
0, 499.66 * .001), JointType.Rotational, true);
            mb_TS.SetJointLimit("J1", 0, 1425 * .001);
            mb_TS.SetJointLimit("J2", -720 * (Math.PI / 180), 720 * (Math.PI / 180));
            Mechanism mech_TS = mb_TS.CompileMechanism();
            station.GraphicComponents.Add(mech_TS);
            mech_TS.Name = "TS with Lift";
            mech_TS.Transform.RZ = 180 * (Math.PI / 180);
        }
    }

    public static void Create()
    {
        //Create a tab
        RibbonTab ribbontab = new RibbonTab("Wolf Positioners", "Wolf Positioners");
        UIEnvironment.RibbonTabs.Add(ribbontab);
        //Make tab active
        UIEnvironment.ActiveRibbonTab = ribbontab;
        //Create a button Group
        RibbonGroup ribbongroup = new RibbonGroup("Part Positioners", "Part Positioners");

        //Create skyhook button
        CommandBarButton button = new CommandBarButton("Create SkyHook");
        button.Caption = "Create SkyHook";
        button.HelpText = "Create a SkyHook";
        button.Enabled = true;
        button.Image = Properties.Resources.SkyHook;
        //Add button event
        button.ExecuteCommand += new ExecuteCommandEventHandler(button_ExecuteCommand);
        //Add button to ribbon group
        ribbongroup.Controls.Add(button);

        //Create skyhook with liftbutton
        CommandBarButton button_SH_Lift = new CommandBarButton("Create SkyHook with Lift");
        button_SH_Lift.Caption = "Create SkyHook with Lift";
        button_SH_Lift.HelpText = "Create a SkyHook with Lift";
```

```
        button_SH_Lift.Enabled = true;
        button_SH_Lift.Image = Properties.Resources.SkyHook_Lift;
        //Add button event
        button_SH_Lift.ExecuteCommand += new
ExecuteCommandEventHandler(button_SH_Lift_ExecuteCommand);
        //Add button to ribbon group
        ribbongroup.Controls.Add(button_SH_Lift);

        //add seperator
        CommandBarSeparator seperator = new CommandBarSeparator();
        ribbongroup.Controls.Add(seperator);

        //Create HSTS Button
        CommandBarButton button2 = new CommandBarButton("Create HSTS");
        button2.Caption = "Create HSTS";
        button2.HelpText = "Create a HSTS";
        button2.Enabled = true;
        button2.Image = Properties.Resources.HSTS;
        //Add button2 event
        button2.ExecuteCommand += new ExecuteCommandEventHandler(button2_ExecuteCommand);
        //Add button2 to ribbon group
        ribbongroup.Controls.Add(button2);

        CommandBarButton button3 = new CommandBarButton("Create HSTS with Lift");
        button3.Caption = "Create HSTS with Lift";
        button3.HelpText = "Create a HSTS with Lift";
        button3.Enabled = true;
        button3.Image = Properties.Resources.HSLIFT;
        //Add button3 event
        button3.ExecuteCommand += new ExecuteCommandEventHandler(button3_ExecuteCommand);
        //Add button3 to ribbon group
        ribbongroup.Controls.Add(button3);

        //add seperator
        ribbongroup.Controls.Add(seperator);

        //Create DC Button
        CommandBarButton DC = new CommandBarButton("Create Drop Center");
        DC.Caption = "Create Drop Center";
        DC.HelpText = "Create Drop Center";
        DC.Enabled = true;
        DC.Image = Properties.Resources.DC;
        //Add button2 event
        DC.ExecuteCommand += new ExecuteCommandEventHandler(DC_ExecuteCommand);
        //Add button2 to ribbon group
        ribbongroup.Controls.Add(DC);

        //Add ribbongroup2 to tab
```

```
ribbontab.Groups.Add(ribbongroup);



//Create a button Group
RibbonGroup ribbongroup2 = new RibbonGroup("Robot Positioner", "Robot Positioner");

//Create TT Button
CommandBarButton TT = new CommandBarButton("Robot Travel Track");
TT.Caption = "Create Robot Travel Track";
TT.HelpText = "Create Robot Travel Track";
TT.Enabled = true;
TT.Image = Properties.Resources.TT;
//Add button2 event
TT.ExecuteCommand += new ExecuteCommandEventHandler(TT_ExecuteCommand);
//Add button2 to ribbon group
ribbongroup2.Controls.Add(TT);

//Add ribbongroup2 to tab
ribbontab.Groups.Add(ribbongroup2);

//Create a button Group
RibbonGroup ribbongroup3 = new RibbonGroup("Saftey Equipment", "Saftey Equipment");

//Create Fence Button
CommandBarButton Fence = new CommandBarButton("Create Fencing");
Fence.Caption = "Create Fencing";
Fence.HelpText = "Create Fencing";
Fence.Enabled = true;
Fence.Image = Properties.Resources.Fence;
//Add button2 event
Fence.ExecuteCommand += new ExecuteCommandEventHandler(Fence_ExecuteCommand);
//Add button2 to ribbon group
ribbongroup3.Controls.Add(Fence);

//add seperator
ribbongroup.Controls.Add(seperator);

//Create Light Button
CommandBarButton Light = new CommandBarButton("Light Curtain");
Light.Caption = "Light Curtain";
Light.HelpText = "Light Curtain";
Light.Enabled = true;
Light.Image = Properties.Resources.Light;
//Add button2 event
Light.ExecuteCommand += new ExecuteCommandEventHandler(Light_ExecuteCommand);
//Add button2 to ribbon group
ribbongroup3.Controls.Add(Light);
```

```
            //Add ribbongroup2 to tab
            ribbontab.Groups.Add(ribbongroup3);

        }

        public static void AddinMain()
        {
            Logger.AddMessage(new LogMessage("Thank you for using the Wolf Positioners"));
            Logger.AddMessage(new LogMessage("Please forward any errors to
Joseph.Kopacz@wolfrobotics.com "));
            Create();
        }
    }
}
```

Form1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Wolf_Positioners_Addin
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
         load.Enabled = false;
      }

      private void button1_Click(object sender, EventArgs e)
      {
         Class1.settings.skyhook_capacity = comboBox1.Text;
         Class1.settings.skyhook_drop = comboBox2.Text;
         Class1.settings.skyhook_throw = comboBox3.Text;

         Class1.Build();
         this.Close();
      }

      private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
      {
         //Clear Previous
         comboBox2.Items.Clear();
         comboBox3.Items.Clear();

         //Build Drop & Throw
         if (comboBox1.Text == "1000Kg")
         {
            comboBox2.Items.AddRange(new object[] { "400", "500", "600", "700", "800", "900", "1000" });
            comboBox3.Items.AddRange(new object[] { "400", "500", "600", "700", "800", "900", "1000",
"1100", "1200" });
         }
         if (comboBox1.Text == "3000Kg")
         {
            comboBox2.Items.AddRange(new object[] { "500", "600", "700", "800", "900", "1000" });
            comboBox3.Items.AddRange(new object[] { "1500", "1600", "1700", "1800", "1900", "2000" });
         }
```

```
        if (comboBox1.Text == "5000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "500", "600", "700", "800", "900", "1000" });
            comboBox3.Items.AddRange(new object[] { "1500", "1600", "1700", "1800", "1900", "2000" });
        }
        if (comboBox1.Text == "10000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "500", "600", "700", "800", "900", "1000" });
            comboBox3.Items.AddRange(new object[] { "1500", "1600", "1700", "1800", "1900", "2000",
"2100", "2200", "2300", "2400", "2500", "2600", "2700", "2800", "2900" });
        }

        if ((comboBox1.Text == String.Empty) || (comboBox2.Text == String.Empty) || (comboBox3.Text
== String.Empty))
        {
            load.Enabled = false;
        }
        if ((comboBox1.Text != String.Empty) && (comboBox2.Text != String.Empty) && (comboBox3.Text
!= String.Empty))
        {
            load.Enabled = true;
        }
    }

    private void cancel_Click(object sender, EventArgs e)
    {
        this.Close();
    }


    private void comboBox2_SelectedIndexChanged_1(object sender, EventArgs e)
    {
        if ((comboBox1.Text == String.Empty) || (comboBox2.Text == String.Empty) || (comboBox3.Text
== String.Empty))
        {
            load.Enabled = false;
        }
        if ((comboBox1.Text != String.Empty) && (comboBox2.Text != String.Empty) && (comboBox3.Text
!= String.Empty))
        {
            load.Enabled = true;
        }
    }

    private void comboBox3_SelectedIndexChanged_1(object sender, EventArgs e)
    {
        if ((comboBox1.Text == String.Empty) || (comboBox2.Text == String.Empty) || (comboBox3.Text
== String.Empty))
```

```
        {
            load.Enabled = false;
        }
        if ((comboBox1.Text != String.Empty) && (comboBox2.Text != String.Empty) && (comboBox3.Text
!= String.Empty))
        {
            load.Enabled = true;
        }
    }

    private void Form1_Load(object sender, EventArgs e)
    {

    }
  }
}
```

Form1.Designer

```
namespace Wolf_Positioners_Addin
{
   partial class Form1
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
         this.load = new System.Windows.Forms.Button();
         this.pictureBox1 = new System.Windows.Forms.PictureBox();
         this.comboBox1 = new System.Windows.Forms.ComboBox();
         this.label1 = new System.Windows.Forms.Label();
         this.comboBox2 = new System.Windows.Forms.ComboBox();
         this.comboBox3 = new System.Windows.Forms.ComboBox();
         this.label2 = new System.Windows.Forms.Label();
         this.label3 = new System.Windows.Forms.Label();
         this.cancel = new System.Windows.Forms.Button();
         ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
         this.SuspendLayout();
         //
         // load
         //
```

```
            this.load.Location = new System.Drawing.Point(185, 237);
            this.load.Margin = new System.Windows.Forms.Padding(2);
            this.load.Name = "load";
            this.load.Size = new System.Drawing.Size(72, 33);
            this.load.TabIndex = 0;
            this.load.Text = "Load";
            this.load.UseVisualStyleBackColor = true;
            this.load.Click += new System.EventHandler(this.button1_Click);
            //
            // pictureBox1
            //
            this.pictureBox1.ErrorImage = null;
            this.pictureBox1.Image = ((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
            this.pictureBox1.Location = new System.Drawing.Point(140, 10);
            this.pictureBox1.Margin = new System.Windows.Forms.Padding(2);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(194, 223);
            this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox1.TabIndex = 1;
            this.pictureBox1.TabStop = false;
            this.pictureBox1.UseWaitCursor = true;
            //
            // comboBox1
            //
            this.comboBox1.FormattingEnabled = true;
            this.comboBox1.Items.AddRange(new object[] {"1000Kg", "2000Kg", "3000Kg", "5000Kg",
"10000Kg"});
            this.comboBox1.Location = new System.Drawing.Point(8, 26);
            this.comboBox1.Margin = new System.Windows.Forms.Padding(2);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(110, 21);
            this.comboBox1.TabIndex = 2;
            this.comboBox1.SelectedIndexChanged += new
System.EventHandler(this.comboBox1_SelectedIndexChanged);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(5, 10);
            this.label1.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(48, 13);
            this.label1.TabIndex = 3;
            this.label1.Text = "Capacity";
            //
            // comboBox2
            //
            this.comboBox2.FormattingEnabled = true;
```

```
        this.comboBox2.Location = new System.Drawing.Point(8, 73);
        this.comboBox2.Margin = new System.Windows.Forms.Padding(2);
        this.comboBox2.Name = "comboBox2";
        this.comboBox2.Size = new System.Drawing.Size(110, 21);
        this.comboBox2.TabIndex = 4;
        this.comboBox2.SelectedIndexChanged += new
System.EventHandler(this.comboBox2_SelectedIndexChanged_1);
        //
        // comboBox3
        //
        this.comboBox3.FormattingEnabled = true;
        this.comboBox3.Location = new System.Drawing.Point(8, 119);
        this.comboBox3.Margin = new System.Windows.Forms.Padding(2);
        this.comboBox3.Name = "comboBox3";
        this.comboBox3.Size = new System.Drawing.Size(110, 21);
        this.comboBox3.TabIndex = 5;
        this.comboBox3.SelectedIndexChanged += new
System.EventHandler(this.comboBox3_SelectedIndexChanged_1);
        //
        // label2
        //
        this.label2.AutoSize = true;
        this.label2.Location = new System.Drawing.Point(5, 57);
        this.label2.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
        this.label2.Name = "label2";
        this.label2.Size = new System.Drawing.Size(37, 13);
        this.label2.TabIndex = 6;
        this.label2.Text = "Throw";
        //
        // label3
        //
        this.label3.AutoSize = true;
        this.label3.Location = new System.Drawing.Point(9, 104);
        this.label3.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
        this.label3.Name = "label3";
        this.label3.Size = new System.Drawing.Size(30, 13);
        this.label3.TabIndex = 7;
        this.label3.Text = "Drop";
        //
        // cancel
        //
        this.cancel.Location = new System.Drawing.Point(262, 237);
        this.cancel.Margin = new System.Windows.Forms.Padding(2);
        this.cancel.Name = "cancel";
        this.cancel.Size = new System.Drawing.Size(72, 33);
        this.cancel.TabIndex = 8;
        this.cancel.Text = "Cancel";
        this.cancel.UseVisualStyleBackColor = true;
```

```
            this.cancel.Click += new System.EventHandler(this.cancel_Click);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(350, 278);
            this.Controls.Add(this.cancel);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.comboBox3);
            this.Controls.Add(this.comboBox2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.comboBox1);
            this.Controls.Add(this.pictureBox1);
            this.Controls.Add(this.load);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.Margin = new System.Windows.Forms.Padding(2);
            this.Name = "Form1";
            this.Text = "Sky Hooks";
            this.Load += new System.EventHandler(this.Form1_Load);
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        public System.Windows.Forms.ComboBox comboBox1;
        public System.Windows.Forms.ComboBox comboBox2;
        public System.Windows.Forms.ComboBox comboBox3;
        public System.Windows.Forms.Button load;
        public System.Windows.Forms.Button cancel;

    }
}
```

Form2

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Wolf_Positioners_Addin
{
   public partial class Form2 : Form
   {
      public Form2()
      {
         InitializeComponent();
         button1.Enabled = false;
      }

      private void Form2_Load(object sender, EventArgs e)
      {
      }

      private void button1_Click(object sender, EventArgs e)
      {
         Class1.settings.HSTS_capacity = comboBox1.Text;
         Class1.settings.HSTS_riser_height = comboBox2.Text;

         Class1.Build2();
         this.Close();
      }

      private void button2_Click(object sender, EventArgs e)
      {
         this.Close();
      }

      private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
      {
         //Clear Previous
         comboBox2.Items.Clear();

         //Build Drop & Throw
         if (comboBox1.Text == "3000Kg")
         {
            comboBox2.Items.AddRange(new object[] { "1000", "1100", "1200", "1300", "1400", "1500",
"1600", "1700", "1800", "1900", "2000"});
```

```
        }
        if (comboBox1.Text == "5000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "1000", "1100", "1200", "1300", "1400", "1500",
"1600", "1700", "1800", "1900", "2000"});
        }
        if (comboBox1.Text == "10000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "1500", "1600", "1700", "1800", "1900", "2000",
"2100", "2200", "2300", "2400", "2500", "2600" });
        }
        if (comboBox1.Text == "20000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "1800", "1900", "2000", "2100", "2200", "2300",
"2400", "2500", "2600", "2700", "2800", "2900", "3000", "3100", "3200", "3300", "3400", "3500" });
        }
        if (comboBox1.Text == "30000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "2000", "2100", "2200", "2300", "2400", "2500",
"2600", "2700", "2800", "2900", "3000"});
        }
        if (comboBox1.Text == "50000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "2600", "2700", "2800", "2900", "3000", "3100",
"3200", "3300", "3400", "3500" });
        }
        if (comboBox1.Text == "70000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "2500", "2600", "2700", "2800", "2900", "3000",
"3100", "3200", "3300", "3400", "3500" });
        }

        if ((comboBox1.Text == String.Empty) || (comboBox2.Text == String.Empty))
        {
            button1.Enabled = false;

        }

        if ((comboBox1.Text != String.Empty) && (comboBox2.Text != String.Empty))
        {
            button1.Enabled = true;
        }
    }

    private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
    {
        if ((comboBox1.Text == String.Empty) || (comboBox2.Text == String.Empty))
        {
```

```
            button1.Enabled = false;
        }
        if ((comboBox1.Text != String.Empty) && (comboBox2.Text != String.Empty))
        {
            button1.Enabled = true;
        }
    }
  }
}
```

Form2.Designer

```
namespace Wolf_Positioners_Addin
{
   partial class Form2
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.button1 = new System.Windows.Forms.Button();
         this.button2 = new System.Windows.Forms.Button();
         this.pictureBox1 = new System.Windows.Forms.PictureBox();
         this.label1 = new System.Windows.Forms.Label();
         this.comboBox1 = new System.Windows.Forms.ComboBox();
         this.label2 = new System.Windows.Forms.Label();
         this.comboBox2 = new System.Windows.Forms.ComboBox();
         ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
         this.SuspendLayout();
         //
         // button1
         //
         this.button1.Location = new System.Drawing.Point(185, 237);
         this.button1.Margin = new System.Windows.Forms.Padding(2);
         this.button1.Name = "button1";
         this.button1.Size = new System.Drawing.Size(72, 33);
```

```
this.button1.TabIndex = 0;
this.button1.Text = "Load";
this.button1.UseVisualStyleBackColor = true;
this.button1.Click += new System.EventHandler(this.button1_Click);
//
// button2
//
this.button2.Location = new System.Drawing.Point(262, 237);
this.button2.Margin = new System.Windows.Forms.Padding(2);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(72, 33);
this.button2.TabIndex = 1;
this.button2.Text = "Cancle";
this.button2.UseVisualStyleBackColor = true;
this.button2.Click += new System.EventHandler(this.button2_Click);
//
// pictureBox1
//
this.pictureBox1.Image = global::Wolf_Positioners_Addin.Properties.Resources.HSTS;
this.pictureBox1.Location = new System.Drawing.Point(140, 10);
this.pictureBox1.Margin = new System.Windows.Forms.Padding(2);
this.pictureBox1.Name = "pictureBox1";
this.pictureBox1.Size = new System.Drawing.Size(194, 223);
this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;
this.pictureBox1.TabIndex = 2;
this.pictureBox1.TabStop = false;
//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(5, 10);
this.label1.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(48, 13);
this.label1.TabIndex = 3;
this.label1.Text = "Capacity";
//
// comboBox1
//
this.comboBox1.FormattingEnabled = true;
this.comboBox1.Items.AddRange(new object[] {
"3000Kg",
"5000Kg",
"10000Kg",
"20000Kg",
"30000Kg",
"50000Kg",
"70000Kg"});
```

```
        this.comboBox1.Location = new System.Drawing.Point(8, 26);
        this.comboBox1.Margin = new System.Windows.Forms.Padding(2);
        this.comboBox1.Name = "comboBox1";
        this.comboBox1.Size = new System.Drawing.Size(110, 21);
        this.comboBox1.TabIndex = 4;
        this.comboBox1.SelectedIndexChanged += new
System.EventHandler(this.comboBox1_SelectedIndexChanged);
        //
        // label2
        //
        this.label2.AutoSize = true;
        this.label2.Location = new System.Drawing.Point(5, 57);
        this.label2.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
        this.label2.Name = "label2";
        this.label2.Size = new System.Drawing.Size(65, 13);
        this.label2.TabIndex = 5;
        this.label2.Text = "Riser Height";
        //
        // comboBox2
        //
        this.comboBox2.FormattingEnabled = true;
        this.comboBox2.Location = new System.Drawing.Point(8, 73);
        this.comboBox2.Margin = new System.Windows.Forms.Padding(2);
        this.comboBox2.Name = "comboBox2";
        this.comboBox2.Size = new System.Drawing.Size(110, 21);
        this.comboBox2.TabIndex = 6;
        this.comboBox2.SelectedIndexChanged += new
System.EventHandler(this.comboBox2_SelectedIndexChanged);
        //
        // Form2
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(350, 278);
        this.Controls.Add(this.comboBox2);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.comboBox1);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.pictureBox1);
        this.Controls.Add(this.button2);
        this.Controls.Add(this.button1);
        this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
        this.Margin = new System.Windows.Forms.Padding(2);
        this.Name = "Form2";
        this.Text = "HSTS";
        this.Load += new System.EventHandler(this.Form2_Load);
        ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
        this.ResumeLayout(false);
```

```
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.ComboBox comboBox1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.ComboBox comboBox2;
    }
}
```

Form3

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Wolf_Positioners_Addin
{
   public partial class Form3 : Form
   {
      public Form3()
      {
         InitializeComponent();
         load.Enabled = false;
      }

      private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
      {
         if ((comboBox1.Text == String.Empty) || (comboBox2.Text == String.Empty) || (comboBox3.Text
== String.Empty))
         {
            load.Enabled = false;
         }
         if ((comboBox1.Text != String.Empty) && (comboBox2.Text != String.Empty) && (comboBox3.Text
!= String.Empty))
         {
            load.Enabled = true;
         }
      }

      private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
      {
         //Clear Previous
         comboBox2.Items.Clear();
         comboBox3.Items.Clear();

         //Build Drop & Throw
         if (comboBox1.Text == "1000Kg")
         {
            comboBox2.Items.AddRange(new object[] { "400", "500", "600", "700", "800", "900", "1000" });
            comboBox3.Items.AddRange(new object[] { "400", "500", "600", "700", "800", "900", "1000",
"1100", "1200" });
         }
         if (comboBox1.Text == "3000Kg")
```

```
        {
            comboBox2.Items.AddRange(new object[] { "500", "600", "700", "800", "900", "1000" });
            comboBox3.Items.AddRange(new object[] { "1500", "1600", "1700", "1800", "1900", "2000" });
        }
        if (comboBox1.Text == "5000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "500", "600", "700", "800", "900", "1000" });
            comboBox3.Items.AddRange(new object[] { "1500", "1600", "1700", "1800", "1900", "2000" });
        }

        if ((comboBox1.Text == String.Empty) || (comboBox2.Text == String.Empty) || (comboBox3.Text
== String.Empty))
        {
            load.Enabled = false;
        }
        if ((comboBox1.Text != String.Empty) && (comboBox2.Text != String.Empty) && (comboBox3.Text
!= String.Empty))
        {
            load.Enabled = true;
        }
    }

    private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
    {
        if ((comboBox1.Text == String.Empty) || (comboBox2.Text == String.Empty) || (comboBox3.Text
== String.Empty))
        {
            load.Enabled = false;
        }
        if ((comboBox1.Text != String.Empty) && (comboBox2.Text != String.Empty) && (comboBox3.Text
!= String.Empty))
        {
            load.Enabled = true;
        }
    }

    private void load_Click(object sender, EventArgs e)
    {
        Class1.settings.skyhook_capacity = comboBox1.Text;
        Class1.settings.skyhook_drop = comboBox2.Text;
        Class1.settings.skyhook_throw = comboBox3.Text;

        Class1.Build3();
        this.Close();
    }

    private void cancel_Click(object sender, EventArgs e)
    {
```

```
            this.Close();
        }

        private void Form3_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Form3.Designer

```
namespace Wolf_Positioners_Addin
{
   partial class Form3
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

 #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.load = new System.Windows.Forms.Button();
         this.pictureBox1 = new System.Windows.Forms.PictureBox();
         this.comboBox1 = new System.Windows.Forms.ComboBox();
         this.label1 = new System.Windows.Forms.Label();
         this.comboBox2 = new System.Windows.Forms.ComboBox();
         this.comboBox3 = new System.Windows.Forms.ComboBox();
         this.label2 = new System.Windows.Forms.Label();
         this.label3 = new System.Windows.Forms.Label();
         this.cancel = new System.Windows.Forms.Button();
         ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
         this.SuspendLayout();
         //
         // load
         //
         this.load.Location = new System.Drawing.Point(185, 237);
         this.load.Margin = new System.Windows.Forms.Padding(2);
```

```
            this.load.Name = "load";
            this.load.Size = new System.Drawing.Size(72, 33);
            this.load.TabIndex = 0;
            this.load.Text = "Load";
            this.load.UseVisualStyleBackColor = true;
            this.load.Click += new System.EventHandler(this.load_Click);
            //
            // pictureBox1
            //
            this.pictureBox1.ErrorImage = null;
            this.pictureBox1.Image = global::Wolf_Positioners_Addin.Properties.Resources.SkyHook_Lift;
            this.pictureBox1.Location = new System.Drawing.Point(140, 10);
            this.pictureBox1.Margin = new System.Windows.Forms.Padding(2);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(194, 223);
            this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox1.TabIndex = 1;
            this.pictureBox1.TabStop = false;
            this.pictureBox1.UseWaitCursor = true;
            //
            // comboBox1
            //
            this.comboBox1.FormattingEnabled = true;
            this.comboBox1.Items.AddRange(new object[] {
            "1000Kg",
            "3000Kg",
            "5000Kg"});
            this.comboBox1.Location = new System.Drawing.Point(8, 27);
            this.comboBox1.Margin = new System.Windows.Forms.Padding(2);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(110, 21);
            this.comboBox1.TabIndex = 2;
            this.comboBox1.SelectedIndexChanged += new
System.EventHandler(this.comboBox1_SelectedIndexChanged);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(5, 10);
            this.label1.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(48, 13);
            this.label1.TabIndex = 3;
            this.label1.Text = "Capacity";
            //
            // comboBox2
            //
            this.comboBox2.FormattingEnabled = true;
```

```
        this.comboBox2.Location = new System.Drawing.Point(8, 74);
        this.comboBox2.Margin = new System.Windows.Forms.Padding(2);
        this.comboBox2.Name = "comboBox2";
        this.comboBox2.Size = new System.Drawing.Size(110, 21);
        this.comboBox2.TabIndex = 4;
        this.comboBox2.SelectedIndexChanged += new
System.EventHandler(this.comboBox2_SelectedIndexChanged);
        //
        // comboBox3
        //
        this.comboBox3.FormattingEnabled = true;
        this.comboBox3.Location = new System.Drawing.Point(8, 121);
        this.comboBox3.Margin = new System.Windows.Forms.Padding(2);
        this.comboBox3.Name = "comboBox3";
        this.comboBox3.Size = new System.Drawing.Size(110, 21);
        this.comboBox3.TabIndex = 5;
        this.comboBox3.SelectedIndexChanged += new
System.EventHandler(this.comboBox3_SelectedIndexChanged);
        //
        // label2
        //
        this.label2.AutoSize = true;
        this.label2.Location = new System.Drawing.Point(5, 57);
        this.label2.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
        this.label2.Name = "label2";
        this.label2.Size = new System.Drawing.Size(37, 13);
        this.label2.TabIndex = 6;
        this.label2.Text = "Throw";
        //
        // label3
        //
        this.label3.AutoSize = true;
        this.label3.Location = new System.Drawing.Point(9, 104);
        this.label3.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
        this.label3.Name = "label3";
        this.label3.Size = new System.Drawing.Size(30, 13);
        this.label3.TabIndex = 7;
        this.label3.Text = "Drop";
        //
        // cancel
        //
        this.cancel.Location = new System.Drawing.Point(262, 237);
        this.cancel.Margin = new System.Windows.Forms.Padding(2);
        this.cancel.Name = "cancel";
        this.cancel.Size = new System.Drawing.Size(72, 33);
        this.cancel.TabIndex = 8;
        this.cancel.Text = "Cancel";
        this.cancel.UseVisualStyleBackColor = true;
```

```
            this.cancel.Click += new System.EventHandler(this.cancel_Click);
            //
            // Form3
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(350, 278);
            this.Controls.Add(this.cancel);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.comboBox3);
            this.Controls.Add(this.comboBox2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.comboBox1);
            this.Controls.Add(this.pictureBox1);
            this.Controls.Add(this.load);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.Margin = new System.Windows.Forms.Padding(2);
            this.Name = "Form3";
            this.Text = "Sky Hooks";
            this.Load += new System.EventHandler(this.Form3_Load);
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        public System.Windows.Forms.ComboBox comboBox1;
        public System.Windows.Forms.ComboBox comboBox2;
        public System.Windows.Forms.ComboBox comboBox3;
        public System.Windows.Forms.Button load;
        public System.Windows.Forms.Button cancel;

    }
}
```

Form4

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Wolf_Positioners_Addin
{
   public partial class Form4 : Form
   {
      public Form4()
      {
         InitializeComponent();
         load.Enabled = false;
      }

      private void Form4_Load(object sender, EventArgs e)
      {

      }

      private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
      {
         //Clear Previous
         comboBox2.Items.Clear();
         comboBox3.Items.Clear();

         //Build Drop & Throw & Update Riser Possibilities
         if (comboBox1.Text == "5000Kg")
         {
            comboBox2.Items.AddRange(new object[] { "3000", "4000", "5000"});
            comboBox4.Items.AddRange(new object[] { "1000", "1100", "1200", "1300", "1400", "1500",
"1600", "1700", "1800", "1900", "2000", "2100", "2200", "2300", "2400", "2500", "2600" });
         }
         if (comboBox1.Text == "10000Kg")
         {
            comboBox2.Items.AddRange(new object[] { "4000", "5000", "6000" });
            comboBox4.Items.AddRange(new object[] { "1500", "1600", "1700", "1800", "1900", "2000",
"2100", "2200", "2300", "2400", "2500", "2600" });
         }
         if (comboBox1.Text == "20000Kg")
         {
            comboBox2.Items.AddRange(new object[] { "6000", "7000", "8000"});
```

```
            comboBox4.Items.AddRange(new object[] { "1800", "1900", "2000", "2100", "2200", "2300",
"2400", "2500", "2600", "2700", "2800", "2900", "3000", "3100", "3200", "3300", "3400", "3500" });
        }
        if (comboBox1.Text == "30000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "7000", "8000", "9000"});
            comboBox4.Items.AddRange(new object[] { "2000", "2100", "2200", "2300", "2400", "2500",
"2600", "2700", "2800", "2900", "3000" });
        }
        if (comboBox1.Text == "50000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "8000", "9000", "10000" });
            comboBox4.Items.AddRange(new object[] { "2600", "2700", "2800", "2900", "3000", "3100",
"3200", "3300", "3400", "3500" });
        }
        if (comboBox1.Text == "70000Kg")
        {
            comboBox2.Items.AddRange(new object[] { "9000", "10000", "11000" });
            comboBox4.Items.AddRange(new object[] { "2500", "2600", "2700", "2800", "2900", "3000",
"3100", "3200", "3300", "3400", "3500" });
        }

        if ((comboBox1.Text == String.Empty) || (comboBox2.Text == String.Empty) || (comboBox3.Text
== String.Empty))
        {
            load.Enabled = false;
        }
        if ((comboBox1.Text != String.Empty) && (comboBox2.Text != String.Empty) && (comboBox3.Text
!= String.Empty))
        {
            load.Enabled = true;
        }
    }


    private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (comboBox1.Text == "5000Kg")
        {
            if (comboBox2.Text == "3000")
            {
                comboBox3.Items.AddRange(new object[] { "200", "300", "400", "500", "600", "700" });
            }
            if (comboBox2.Text == "4000")
            {
                comboBox3.Items.AddRange(new object[] { "300", "400", "500", "600", "700", "800" });
            }
            if (comboBox2.Text == "5000")
```

```
    {
        comboBox3.Items.AddRange(new object[] { "400", "500", "600", "700", "800", "900" });
    }
}

if (comboBox1.Text == "10000Kg")
{
    if (comboBox2.Text == "4000")
    {
        comboBox3.Items.AddRange(new object[] { "300", "400", "500", "600", "700", "800" });
    }
    if (comboBox2.Text == "5000")
    {
        comboBox3.Items.AddRange(new object[] { "400", "500", "600", "700", "800", "900" });
    }
    if (comboBox2.Text == "6000")
    {
        comboBox3.Items.AddRange(new object[] { "500", "600", "700", "800", "900", "1000" });
    }
}

if (comboBox1.Text == "20000Kg")
{
    if (comboBox2.Text == "6000")
    {
        comboBox3.Items.AddRange(new object[] { "400", "500", "600", "700", "800", "900" });
    }
    if (comboBox2.Text == "7000")
    {
        comboBox3.Items.AddRange(new object[] { "500", "600", "700", "800", "900", "1000" });
    }
    if (comboBox2.Text == "8000")
    {
        comboBox3.Items.AddRange(new object[] { "600", "700", "800", "900", "1000", "1100" });
    }
}

if (comboBox1.Text == "30000Kg")
{
    if (comboBox2.Text == "7000")
    {
        comboBox3.Items.AddRange(new object[] { "500", "600", "700", "800", "900", "1000" });
    }
    if (comboBox2.Text == "8000")
    {
        comboBox3.Items.AddRange(new object[] { "600", "700", "800", "900", "1000", "1100" });
    }
    if (comboBox2.Text == "9000")
```

```
            {
                comboBox3.Items.AddRange(new object[] { "700", "800", "900", "1000", "1100", "1200" });
            }
        }

        if (comboBox1.Text == "50000Kg")
        {
            if (comboBox2.Text == "8000")
            {
                comboBox3.Items.AddRange(new object[] { "600", "700", "800", "900", "1000", "1100" });
            }
            if (comboBox2.Text == "9000")
            {
                comboBox3.Items.AddRange(new object[] { "700", "800", "900", "1000", "1100", "1200" });
            }
            if (comboBox2.Text == "10000")
            {
                comboBox3.Items.AddRange(new object[] { "800", "900", "1000", "1100", "1200", "1300" });
            }
        }

        if (comboBox1.Text == "70000Kg")
        {
            if (comboBox2.Text == "9000")
            {
                comboBox3.Items.AddRange(new object[] { "700", "800", "900", "1000", "1100", "1200" });
            }
            if (comboBox2.Text == "10000")
            {
                comboBox3.Items.AddRange(new object[] { "800", "900", "1000", "1100", "1200", "1300" });
            }
            if (comboBox2.Text == "11000")
            {
                comboBox3.Items.AddRange(new object[] { "900", "1000", "1100", "1200", "1300", "1400" });
            }
        }

        if ((comboBox1.Text == String.Empty) || (comboBox2.Text == String.Empty) || (comboBox3.Text
== String.Empty))
        {
            load.Enabled = false;
        }
        if ((comboBox1.Text != String.Empty) && (comboBox2.Text != String.Empty) && (comboBox3.Text
!= String.Empty))
        {
            load.Enabled = true;
        }
    }
```

```
        private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
        {
            if ((comboBox1.Text == String.Empty) || (comboBox2.Text == String.Empty) || (comboBox3.Text
== String.Empty))
            {
                load.Enabled = false;
            }
            if ((comboBox1.Text != String.Empty) && (comboBox2.Text != String.Empty) && (comboBox3.Text
!= String.Empty))
            {
                load.Enabled = true;
            }
        }

        private void load_Click(object sender, EventArgs e)
        {
            Class1.settings.skyhook_capacity = comboBox1.Text;
            Class1.settings.skyhook_throw = comboBox2.Text;
            Class1.settings.skyhook_drop = comboBox3.Text;
            Class1.settings.DC_riser_height = comboBox4.Text;

            Class1.Build4();
            this.Close();
        }

        private void cancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void Form4_Load_1(object sender, EventArgs e)
        {

        }
    }
}
```

Form4.Designer

```
namespace Wolf_Positioners_Addin
{
   partial class Form4
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

     #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.load = new System.Windows.Forms.Button();
         this.pictureBox1 = new System.Windows.Forms.PictureBox();
         this.comboBox1 = new System.Windows.Forms.ComboBox();
         this.label1 = new System.Windows.Forms.Label();
         this.comboBox2 = new System.Windows.Forms.ComboBox();
         this.comboBox3 = new System.Windows.Forms.ComboBox();
         this.label2 = new System.Windows.Forms.Label();
         this.label3 = new System.Windows.Forms.Label();
         this.cancel = new System.Windows.Forms.Button();
         this.label4 = new System.Windows.Forms.Label();
         this.comboBox4 = new System.Windows.Forms.ComboBox();
         ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
         this.SuspendLayout();
         //
         // load
         //
```

```
        this.load.Location = new System.Drawing.Point(185, 237);
        this.load.Margin = new System.Windows.Forms.Padding(2);
        this.load.Name = "load";
        this.load.Size = new System.Drawing.Size(72, 33);
        this.load.TabIndex = 0;
        this.load.Text = "Load";
        this.load.UseVisualStyleBackColor = true;
        this.load.Click += new System.EventHandler(this.load_Click);
        //
        // pictureBox1
        //
        this.pictureBox1.ErrorImage = null;
        this.pictureBox1.Image = global::Wolf_Positioners_Addin.Properties.Resources.DC;
        this.pictureBox1.Location = new System.Drawing.Point(140, 10);
        this.pictureBox1.Margin = new System.Windows.Forms.Padding(2);
        this.pictureBox1.Name = "pictureBox1";
        this.pictureBox1.Size = new System.Drawing.Size(194, 130);
        this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;
        this.pictureBox1.TabIndex = 1;
        this.pictureBox1.TabStop = false;
        this.pictureBox1.UseWaitCursor = true;
        //
        // comboBox1
        //
        this.comboBox1.FormattingEnabled = true;
        this.comboBox1.Items.AddRange(new object[] {
        "5000Kg",
        "10000Kg",
        "20000Kg",
        "30000Kg",
        "50000Kg",
        "70000Kg"});
        this.comboBox1.Location = new System.Drawing.Point(8, 26);
        this.comboBox1.Margin = new System.Windows.Forms.Padding(2);
        this.comboBox1.Name = "comboBox1";
        this.comboBox1.Size = new System.Drawing.Size(110, 21);
        this.comboBox1.TabIndex = 2;
        this.comboBox1.SelectedIndexChanged += new
System.EventHandler(this.comboBox1_SelectedIndexChanged);
        //
        // label1
        //
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(5, 10);
        this.label1.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(48, 13);
        this.label1.TabIndex = 3;
```

```
            this.label1.Text = "Capacity";
            //
            // comboBox2
            //
            this.comboBox2.FormattingEnabled = true;
            this.comboBox2.Location = new System.Drawing.Point(8, 73);
            this.comboBox2.Margin = new System.Windows.Forms.Padding(2);
            this.comboBox2.Name = "comboBox2";
            this.comboBox2.Size = new System.Drawing.Size(110, 21);
            this.comboBox2.TabIndex = 4;
            this.comboBox2.SelectedIndexChanged += new
System.EventHandler(this.comboBox2_SelectedIndexChanged);
            //
            // comboBox3
            //
            this.comboBox3.FormattingEnabled = true;
            this.comboBox3.Location = new System.Drawing.Point(8, 119);
            this.comboBox3.Margin = new System.Windows.Forms.Padding(2);
            this.comboBox3.Name = "comboBox3";
            this.comboBox3.Size = new System.Drawing.Size(110, 21);
            this.comboBox3.TabIndex = 5;
            this.comboBox3.SelectedIndexChanged += new
System.EventHandler(this.comboBox3_SelectedIndexChanged);
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(5, 57);
            this.label2.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(37, 13);
            this.label2.TabIndex = 6;
            this.label2.Text = "Throw";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(5, 104);
            this.label3.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(30, 13);
            this.label3.TabIndex = 7;
            this.label3.Text = "Drop";
            //
            // cancel
            //
            this.cancel.Location = new System.Drawing.Point(262, 237);
            this.cancel.Margin = new System.Windows.Forms.Padding(2);
```

```
this.cancel.Name = "cancel";
this.cancel.Size = new System.Drawing.Size(72, 33);
this.cancel.TabIndex = 8;
this.cancel.Text = "Cancel";
this.cancel.UseVisualStyleBackColor = true;
this.cancel.Click += new System.EventHandler(this.cancel_Click);
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(5, 150);
this.label4.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(65, 13);
this.label4.TabIndex = 10;
this.label4.Text = "Riser Height";
//
// comboBox4
//
this.comboBox4.FormattingEnabled = true;
this.comboBox4.Location = new System.Drawing.Point(8, 165);
this.comboBox4.Margin = new System.Windows.Forms.Padding(2);
this.comboBox4.Name = "comboBox4";
this.comboBox4.Size = new System.Drawing.Size(110, 21);
this.comboBox4.TabIndex = 9;
//
// Form4
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(350, 278);
this.Controls.Add(this.label4);
this.Controls.Add(this.comboBox4);
this.Controls.Add(this.cancel);
this.Controls.Add(this.label3);
this.Controls.Add(this.label2);
this.Controls.Add(this.comboBox3);
this.Controls.Add(this.comboBox2);
this.Controls.Add(this.label1);
this.Controls.Add(this.comboBox1);
this.Controls.Add(this.pictureBox1);
this.Controls.Add(this.load);
this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
this.Margin = new System.Windows.Forms.Padding(2);
this.Name = "Form4";
this.Text = "Drop Center";
this.Load += new System.EventHandler(this.Form4_Load_1);
((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
```

```
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        public System.Windows.Forms.ComboBox comboBox1;
        public System.Windows.Forms.ComboBox comboBox2;
        public System.Windows.Forms.ComboBox comboBox3;
        public System.Windows.Forms.Button load;
        public System.Windows.Forms.Button cancel;
        private System.Windows.Forms.Label label4;
        public System.Windows.Forms.ComboBox comboBox4;

    }
}
```

Form5

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Wolf_Positioners_Addin
{
   public partial class Form5 : Form
   {
      public Form5()
      {
         InitializeComponent();
         button1.Enabled = false;
      }

      private void Form2_Load(object sender, EventArgs e)
      {
      }

      private void button1_Click(object sender, EventArgs e)
      {
         Class1.settings.HSTS_capacity = comboBox1.Text;

         Class1.Build5();
         this.Close();
      }

      private void button2_Click(object sender, EventArgs e)
      {
         this.Close();
      }

      private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
      {
         //Clear Previous


         if ((comboBox1.Text == String.Empty))
         {
            button1.Enabled = false;

         }
```

```
            if ((comboBox1.Text != String.Empty))
            {
                button1.Enabled = true;
            }
        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            if ((comboBox1.Text == String.Empty))
            {
                button1.Enabled = false;
            }
            if ((comboBox1.Text != String.Empty))
            {
                button1.Enabled = true;
            }
        }
    }
}
```

Form5.Designer

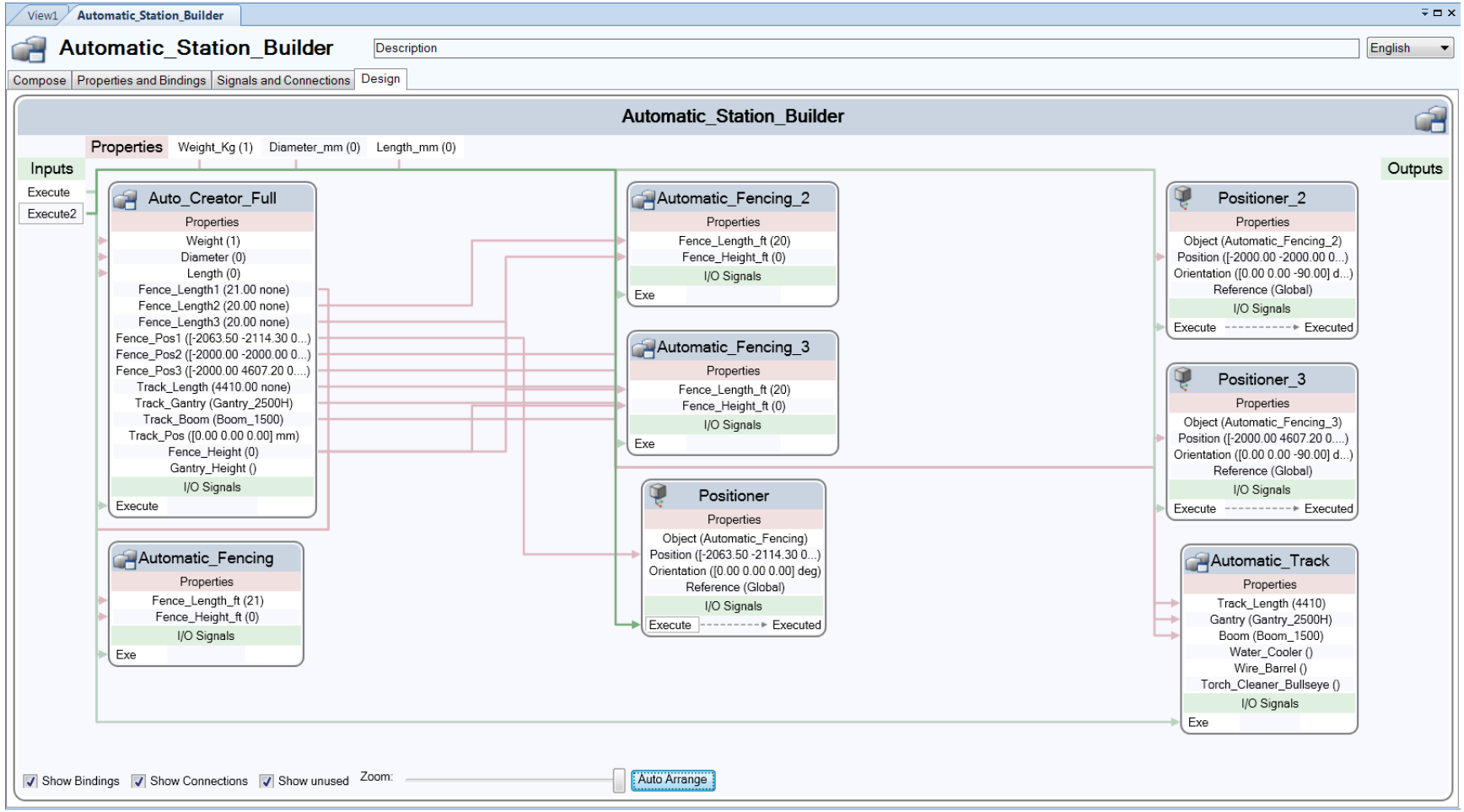
```
namespace Wolf_Positioners_Addin
{
   partial class Form5
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.button1 = new System.Windows.Forms.Button();
         this.button2 = new System.Windows.Forms.Button();
         this.label1 = new System.Windows.Forms.Label();
         this.comboBox1 = new System.Windows.Forms.ComboBox();
         this.pictureBox1 = new System.Windows.Forms.PictureBox();
         ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
         this.SuspendLayout();
         //
         // button1
         //
         this.button1.Location = new System.Drawing.Point(185, 237);
         this.button1.Margin = new System.Windows.Forms.Padding(2);
         this.button1.Name = "button1";
         this.button1.Size = new System.Drawing.Size(72, 33);
         this.button1.TabIndex = 0;
         this.button1.Text = "Load";
```

```
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.Location = new System.Drawing.Point(262, 237);
            this.button2.Margin = new System.Windows.Forms.Padding(2);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(72, 33);
            this.button2.TabIndex = 1;
            this.button2.Text = "Cancle";
            this.button2.UseVisualStyleBackColor = true;
            this.button2.Click += new System.EventHandler(this.button2_Click);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(5, 10);
            this.label1.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(48, 13);
            this.label1.TabIndex = 3;
            this.label1.Text = "Capacity";
            //
            // comboBox1
            //
            this.comboBox1.FormattingEnabled = true;
            this.comboBox1.Items.AddRange(new object[] {
            "1000Kg",
            "3000Kg",
            "5000Kg"});
            this.comboBox1.Location = new System.Drawing.Point(8, 26);
            this.comboBox1.Margin = new System.Windows.Forms.Padding(2);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(110, 21);
            this.comboBox1.TabIndex = 4;
            this.comboBox1.SelectedIndexChanged += new
System.EventHandler(this.comboBox1_SelectedIndexChanged);
            //
            // pictureBox1
            //
            this.pictureBox1.Image = global::Wolf_Positioners_Addin.Properties.Resources.HSLIFT;
            this.pictureBox1.Location = new System.Drawing.Point(140, 10);
            this.pictureBox1.Margin = new System.Windows.Forms.Padding(2);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(194, 223);
            this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox1.TabIndex = 2;
```

```
        this.pictureBox1.TabStop = false;
        //
        // Form5
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(350, 278);
        this.Controls.Add(this.comboBox1);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.pictureBox1);
        this.Controls.Add(this.button2);
        this.Controls.Add(this.button1);
        this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
        this.Margin = new System.Windows.Forms.Padding(2);
        this.Name = "Form5";
        this.Text = "HSTS";
        this.Load += new System.EventHandler(this.Form2_Load);
        ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
        this.ResumeLayout(false);
        this.PerformLayout();

    }

    #endregion

    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.Button button2;
    private System.Windows.Forms.PictureBox pictureBox1;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.ComboBox comboBox1;
}
```

APPENDIX E. SOURCE CODE AUTOMATIC STATION BUILDER

RobotStudio



CodeBehind

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.Text;
using System.IO;

using ABB.Robotics.Math;
using ABB.Robotics.RobotStudio;
using ABB.Robotics.RobotStudio.Stations;

namespace Auto_Creator_Full
{
   /// <summary>
   /// Code-behind class for the Auto_Creator_Full Smart Component.
   /// </summary>
   /// <remarks>
   /// The code-behind class should be seen as a service provider used by the
   /// Smart Component runtime. Only one instance of the code-behind class
   /// is created, regardless of how many instances there are of the associated
   /// Smart Component.
   /// Therefore, the code-behind class should not store any state information.
   /// Instead, use the SmartComponent.StateCache collection.
   /// </remarks>
   public class CodeBehind : SmartComponentCodeBehind
   {
      public static class settings
      {
         public static string HSTS_capacity = "500";
         public static string HSTS_riser_height = "500";
         public static int RTT = 500;
         public static double Curtain_Transform_Y = 100;
         public static double Curtain_Transform_X = 100;
         public static double Curtain_Len = 100;
      }

      /// <summary>
      /// Called when the value of a dynamic property value has changed.
      /// </summary>
      /// <param name="component"> Component that owns the changed property. </param>
      /// <param name="changedProperty"> Changed property. </param>
      /// <param name="oldValue"> Previous value of the changed property. </param>
      //public override void OnPropertyValueChanged(SmartComponent component, DynamicProperty
changedProperty, object oldValue)
      public override void OnIOSignalValueChanged(SmartComponent component, IOSignal
changedSignal)
      {
```

```
        string Orig_Weight = Convert.ToString(component.Properties["Weight"].Value);
        string Orig_Diam = Convert.ToString(component.Properties["Diameter"].Value);
        string Orig_Length = Convert.ToString(component.Properties["Length"].Value);

        //Fix Units
        //.ToString(Convert.ToInt32(Orig_Diam * ));
        //Orig_Length = Convert.ToString(Convert.ToInt32(Orig_Length) * 1000);


        //Remove Old
        string[] to_delet = { "Head Stock", "Head Stock Riser", "Tail Stock", "Tail Stock Riser",
"Curtain_8ft_0", "Light_8ft_0", "Curtain_12ft_0", "Light_12ft_0" };
        Station station = Project.ActiveProject as Station;
        for (int i = 0; i < 8; i++)
        {
          GraphicComponent to_del;
          station.GraphicComponents.TryGetGraphicComponent(to_delet[i], out to_del);

          if (to_del != null)
          {
            station.GraphicComponents.Remove(to_del);
          }
        }



        //Select HSTS
        if (Convert.ToInt32(Orig_Weight) >= 0 & Convert.ToInt32(Orig_Weight) < 3000)
        {
          CodeBehind.settings.HSTS_capacity = "3000Kg";

          //Select Riser
          string[] temp = { "0", "1000", "1100", "1200", "1300", "1400", "1500", "1600", "1700", "1800",
"1900", "2000", "2100", "2200", "2300", "2400", "2500" };
          for (int i = 0; i < 15; i++)
          {
            if (Convert.ToInt32(Orig_Diam)/2 >= Convert.ToInt32(temp[i]) &
Convert.ToInt32(Orig_Diam)/2 < Convert.ToInt32(temp[i + 1]))
            {
              CodeBehind.settings.HSTS_riser_height = temp[i + 1];
            }
          }
        }

        if (Convert.ToInt32(Orig_Weight) >= 3000 & Convert.ToInt32(Orig_Weight) < 5000)
        {
          CodeBehind.settings.HSTS_capacity = "5000Kg";
```

```
            //Select Riser
            string[] temp = { "0", "1000", "1100", "1200", "1300", "1400", "1500", "1600", "1700", "1800",
"1900", "2000", "2100", "2200", "2300", "2400", "2500", "2600" };
            for (int i = 0; i < 16; i++)
            {
                if (Convert.ToInt32(Orig_Diam)/2 >= Convert.ToInt32(temp[i]) &
Convert.ToInt32(Orig_Diam)/2 < Convert.ToInt32(temp[i + 1]))
                {
                    CodeBehind.settings.HSTS_riser_height = temp[i + 1];
                }
            }
        }

        if (Convert.ToInt32(Orig_Weight) >= 5000 & Convert.ToInt32(Orig_Weight) < 10000)
        {
            CodeBehind.settings.HSTS_capacity = "10000Kg";

            //Select Riser
            string[] temp = { "0", "1500", "1600", "1700", "1800", "1900", "2000", "2100", "2200", "2300",
"2400", "2500", "2600" };
            for (int i = 0; i < 11; i++)
            {
                if (Convert.ToInt32(Orig_Diam)/2 >= Convert.ToInt32(temp[i]) &
Convert.ToInt32(Orig_Diam)/2 < Convert.ToInt32(temp[i + 1]))
                {
                    CodeBehind.settings.HSTS_riser_height = temp[i + 1];
                }
            }
        }

        if (Convert.ToInt32(Orig_Weight) >= 10000 & Convert.ToInt32(Orig_Weight) < 20000)
        {
            CodeBehind.settings.HSTS_capacity = "20000Kg";

            //Select Riser
            string[] temp = { "0", "1800", "1900", "2000", "2100", "2200", "2300", "2400", "2500", "2600",
"2700", "2800", "2900", "3000", "3100", "3200", "3300", "3400", "3500" };
            for (int i = 0; i < 17; i++)
            {
                if (Convert.ToInt32(Orig_Diam)/2 >= Convert.ToInt32(temp[i]) &
Convert.ToInt32(Orig_Diam)/2 < Convert.ToInt32(temp[i + 1]))
                {
                    CodeBehind.settings.HSTS_riser_height = temp[i + 1];
                }
            }
        }
```

```
        if (Convert.ToInt32(Orig_Weight) >= 20000 & Convert.ToInt32(Orig_Weight) < 30000)
        {
            CodeBehind.settings.HSTS_capacity = "30000Kg";

            //Select Riser
            string[] temp = { "0", "2000", "2100", "2200", "2300", "2400", "2500", "2600", "2700", "2800",
"2900", "3000" };
            for (int i = 0; i < 10; i++)
            {
                if (Convert.ToInt32(Orig_Diam)/2 >= Convert.ToInt32(temp[i]) &
Convert.ToInt32(Orig_Diam)/2 < Convert.ToInt32(temp[i + 1]))
                {
                    CodeBehind.settings.HSTS_riser_height = temp[i + 1];
                }
            }
        }

        if (Convert.ToInt32(Orig_Weight) >= 30000 & Convert.ToInt32(Orig_Weight) < 50000)
        {
            CodeBehind.settings.HSTS_capacity = "50000Kg";

            //Select Riser
            string[] temp = { "0", "2600", "2700", "2800", "2900", "3000", "3100", "3200", "3300", "3400",
"3500" };
            for (int i = 0; i < 9; i++)
            {
                if (Convert.ToInt32(Orig_Diam)/2 >= Convert.ToInt32(temp[i]) &
Convert.ToInt32(Orig_Diam)/2 < Convert.ToInt32(temp[i + 1]))
                {
                    CodeBehind.settings.HSTS_riser_height = temp[i + 1];
                }
            }
        }

        if (Convert.ToInt32(Orig_Weight) >= 50000 & Convert.ToInt32(Orig_Weight) < 70000)
        {
            CodeBehind.settings.HSTS_capacity = "70000Kg";

            //Select Riser
            string[] temp = { "0", "2500", "2600", "2700", "2800", "2900", "3000", "3100", "3200", "3300",
"3400", "3500" };
            for (int i = 0; i < 10; i++)
            {
                if (Convert.ToInt32(Orig_Diam)/2 >= Convert.ToInt32(temp[i]) &
Convert.ToInt32(Orig_Diam)/2 < Convert.ToInt32(temp[i + 1]))
                {
                    CodeBehind.settings.HSTS_riser_height = temp[i + 1];
                }
```

```
    }
}

//Select Gantry Height and Fence Height
if (Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) < 2500)
{
    component.Properties["Track_Gantry"].Value = ("Gantry_2500H");
    component.Properties["Fence_Height"].Value = (8);
}

if (Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) >= 2500)
{
    component.Properties["Track_Gantry"].Value = ("Gantry_3500H");
    component.Properties["Fence_Height"].Value = (12);
}

//Select Robot Travel Track
if (Convert.ToInt32(Orig_Length) > 0 & Convert.ToInt32(Orig_Length) < 4410)
{
    CodeBehind.settings.RTT = 4410;
}

if (Convert.ToInt32(Orig_Length) >= 4410 & Convert.ToInt32(Orig_Length) < 5880)
{
    CodeBehind.settings.RTT = 5880;
}

if (Convert.ToInt32(Orig_Length) >= 5880 & Convert.ToInt32(Orig_Length) < 7350)
{
    CodeBehind.settings.RTT = 7350;
}

if (Convert.ToInt32(Orig_Length) >= 7350 & Convert.ToInt32(Orig_Length) < 8820)
{
    CodeBehind.settings.RTT = 8820;
}

if (Convert.ToInt32(Orig_Length) >= 8820 & Convert.ToInt32(Orig_Length) < 10290)
{
    CodeBehind.settings.RTT = 10290;
}

if (Convert.ToInt32(Orig_Length) >= 10290 & Convert.ToInt32(Orig_Length) < 11760)
{
    CodeBehind.settings.RTT = 11760;
}

if (Convert.ToInt32(Orig_Length) >= 11760 & Convert.ToInt32(Orig_Length) < 13230)
```

```
{
   CodeBehind.settings.RTT = 13230;
}

if (Convert.ToInt32(Orig_Length) >= 13230 & Convert.ToInt32(Orig_Length) < 14700)
{
   CodeBehind.settings.RTT = 14700;
}

if (Convert.ToInt32(Orig_Length) >= 14700 & Convert.ToInt32(Orig_Length) < 16170)
{
   CodeBehind.settings.RTT = 16170;
}

if (Convert.ToInt32(Orig_Length) >= 16170 & Convert.ToInt32(Orig_Length) < 17640)
{
   CodeBehind.settings.RTT = 17640;
}

if (Convert.ToInt32(Orig_Length) >= 17640 & Convert.ToInt32(Orig_Length) < 19110)
{
   CodeBehind.settings.RTT = 19110;
}

if (Convert.ToInt32(Orig_Length) >= 19110 & Convert.ToInt32(Orig_Length) < 20580)
{
   CodeBehind.settings.RTT = 20580;
}

if (Convert.ToInt32(Orig_Length) >= 20580 & Convert.ToInt32(Orig_Length) < 22050)
{
   CodeBehind.settings.RTT = 22050;
}

if (Convert.ToInt32(Orig_Length) >= 22050 & Convert.ToInt32(Orig_Length) < 23520)
{
   CodeBehind.settings.RTT = 23520;
}

if (Convert.ToInt32(Orig_Length) >= 23520 & Convert.ToInt32(Orig_Length) < 24990)
{
   CodeBehind.settings.RTT = 24990;
}

if (Convert.ToInt32(Orig_Length) >= 24990 & Convert.ToInt32(Orig_Length) < 26460)
{
   CodeBehind.settings.RTT = 26460;
}
```

```
        if (Convert.ToInt32(Orig_Length) >= 26460 & Convert.ToInt32(Orig_Length) < 27930)
        {
           CodeBehind.settings.RTT = 27930;
        }

        if (Convert.ToInt32(Orig_Length) >= 27930 & Convert.ToInt32(Orig_Length) < 29400)
        {
           CodeBehind.settings.RTT = 29400;
        }

        if (Convert.ToInt32(Orig_Length) >= 29400 & Convert.ToInt32(Orig_Length) < 30870)
        {
           CodeBehind.settings.RTT = 30870;
        }

        if (Convert.ToInt32(Orig_Length) >= 30870 & Convert.ToInt32(Orig_Length) < 32340)
        {
           CodeBehind.settings.RTT = 32340;
        }

        //Load HSTS
        //Station station = Project.ActiveProject as Station;
        GraphicComponentLibrary mylib;

        //Load HS
        System.Text.StringBuilder file_HS = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\WTU Headstocks\\Wolf_");
        string Path_HS;
        string name_HS = "Wolf_" + settings.HSTS_capacity.ToString() + "_HS";
        Path_HS = file_HS.ToString() + settings.HSTS_capacity.ToString() + "_HS.rslib";
        mylib = GraphicComponentLibrary.Load(Path_HS, true);
        Mechanism HS = (Mechanism)mylib.RootComponent.CopyInstance();
        station.GraphicComponents.Add(HS);
        HS.DisconnectFromLibrary();
        HS.Name = "Head Stock";

        //Load HS Riser
        System.Text.StringBuilder file_HS_Riser = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\Riser\\WTU");
        string Path_HS_Riser;
        string HS_cap = settings.HSTS_capacity.ToString();
        HS_cap = HS_cap.TrimEnd('g');
        HS_cap = HS_cap.TrimEnd('K');
        string name_HS_Riser = "Riser_WTU" + HS_cap + "_" + settings.HSTS_riser_height + "H";
        Path_HS_Riser = file_HS_Riser.ToString() + settings.HSTS_capacity + " Risers\\" + name_HS_Riser
+ ".rslib";
        mylib = GraphicComponentLibrary.Load(Path_HS_Riser, true);
```

```
        station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
        GraphicComponent HS_Riser;
        station.GraphicComponents.TryGetGraphicComponent(name_HS_Riser, out HS_Riser);
        HS_Riser.DisconnectFromLibrary();
        HS_Riser.Transform.RY = (-90 * (Math.PI / 180));
        HS_Riser.Transform.RZ = (180 * (Math.PI / 180));
        HS_Riser.Name = "Head Stock Riser";

        if (settings.HSTS_capacity == "3000Kg" | settings.HSTS_capacity == "5000Kg")
        {
           HS_Riser.Transform.Z = (569.91) * .001;
        }
        if (settings.HSTS_capacity == "10000Kg")
        {
           HS_Riser.Transform.Z = (647.48) * .001;
        }
        if (settings.HSTS_capacity == "20000Kg")
        {
           HS_Riser.Transform.Z = (1105.97) * .001;
        }
        if (settings.HSTS_capacity == "30000Kg")
        {
           HS_Riser.Transform.Z = (1197.87) * .001;
        }
        if (settings.HSTS_capacity == "50000Kg")
        {
           HS_Riser.Transform.RY = 0;
           HS_Riser.Transform.RZ = (90 * (Math.PI / 180));
           HS_Riser.Transform.X = (-1093.50) * .001;
        }
        if (settings.HSTS_capacity == "70000Kg")
        {
           HS_Riser.Transform.Z = (1514.04) * .001;
        }

        //Attach Riser to HS
        GraphicComponent L1;
        HS.GetParentLink(0, out L1);
        Part attacher = (Part)L1;
        attacher.Attach(HS_Riser, false, new Matrix4(new Vector3(0, 0, 0)));

        //Load TS
        if (settings.HSTS_capacity == "3000Kg" | settings.HSTS_capacity == "5000Kg")
        {
           System.Text.StringBuilder file_TS = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\WTU
Tailstocks\\Wolf_3000,5000TS\\WOLF_3000-5000TS_Solid_");
           string Path_TS;
```

```
            string name_TS = "TS-" + settings.HSTS_riser_height;
            Path_TS = file_TS.ToString() + settings.HSTS_riser_height.ToString() + ".rslib";
            mylib = GraphicComponentLibrary.Load(Path_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent TS;
            station.GraphicComponents.TryGetGraphicComponent(name_TS, out TS);
            TS.DisconnectFromLibrary();
            TS.Name = "Tail Stock";
            TS.Transform.RZ = -(3.14 / 180) * 90;
        }
        else
        {
            System.Text.StringBuilder file_TS = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\WTU Tailstocks\\");
            string Path_TS;
            string temp2 = settings.HSTS_capacity.ToString();
            temp2 = temp2.TrimEnd('g');
            temp2 = temp2.TrimEnd('K');
            string name_TS = "Wolf_" + temp2 + "TS";
            Path_TS = file_TS.ToString() + name_TS + ".rslib";
            mylib = GraphicComponentLibrary.Load(Path_TS, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent TS;
            station.GraphicComponents.TryGetGraphicComponent(name_TS, out TS);
            TS.DisconnectFromLibrary();
            TS.Name = "Tail Stock";
        }

        //Load TS Riser
        if (settings.HSTS_capacity == "3000Kg" | settings.HSTS_capacity == "5000Kg")
        {
        }
        else
        {
            System.Text.StringBuilder file_TS_Riser = new System.Text.StringBuilder("C:\\Program Files
(x86)\\Common Files\\ABB Industrial IT\\Robotics IT\\RobotStudio\\Addins\\Riser\\WTU");
            string Path_TS_Riser;
            string temp3 = settings.HSTS_capacity.ToString();
            temp3 = temp3.TrimEnd('g');
            temp3 = temp3.TrimEnd('K');
            string name_TS_Riser = "Riser_WTU" + temp3 + "_" + settings.HSTS_riser_height + "H";
            Path_TS_Riser = file_TS_Riser.ToString() + settings.HSTS_capacity + " Risers\\" + name_TS_Riser
+ ".rslib";
            mylib = GraphicComponentLibrary.Load(Path_HS_Riser, true);
            station.GraphicComponents.Add(mylib.RootComponent.CopyInstance());
            GraphicComponent TS_Riser;
            station.GraphicComponents.TryGetGraphicComponent(name_TS_Riser, out TS_Riser);
            TS_Riser.DisconnectFromLibrary();
```

```
    TS_Riser.Transform.RY = (-90 * (Math.PI / 180));
    TS_Riser.Name = "Tail Stock Riser";

    if (settings.HSTS_capacity == "10000Kg")
    {
        TS_Riser.Transform.Z = (647.48) * .001;
    }
    if (settings.HSTS_capacity == "20000Kg")
    {
        TS_Riser.Transform.Z = (1105.97) * .001;
    }
    if (settings.HSTS_capacity == "30000Kg")
    {
        TS_Riser.Transform.Z = (1197.87) * .001;
    }
    if (settings.HSTS_capacity == "50000Kg")
    {
        TS_Riser.Transform.RY = 0;
        TS_Riser.Transform.RZ = (90 * (Math.PI / 180));
        TS_Riser.Transform.X = (+1093.50) * .001;
    }
    if (settings.HSTS_capacity == "70000Kg")
    {
        TS_Riser.Transform.Z = (1514.04) * .001;
    }

    //Name Tail Stock
    TS_Riser.Name = "Tail Stock Riser";

    //Attach Riser to TS
    GraphicComponent L2;
    station.GraphicComponents.TryGetGraphicComponent("Tail Stock", out L2);
    Part attacher2 = (Part)L2;
    attacher2.Attach(TS_Riser, false, new Matrix4(new Vector3(0, 0, 0)));
    L2.Transform.RZ = -(3.14 / 180) * 90;
    L2.Transform.Z = (Convert.ToInt32(settings.HSTS_riser_height)) * .001;
}


//Place Head Stock based on Part Length and Diameter
HS.Transform.Z = (Convert.ToInt32(settings.HSTS_riser_height)) * .001;
HS.Transform.RZ = (3.14 / 180) * 90;
HS.Transform.X = (1500 + (Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) / 2)) * .001;
HS.Transform.Y = (-1000 + CodeBehind.settings.RTT) * .001;

//Place Tail Stock based on Part Length and Diameter
GraphicComponent TS_move;
station.GraphicComponents.TryGetGraphicComponent("Tail Stock", out TS_move);
```

```
        TS_move.Transform.X = (1500 + (Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) / 2)) *
.001;
        TS_move.Transform.Y = (-1000) * .001;

        //Set Travel Track, Gantry, Boom
        component.Properties["Track_Length"].Value = (CodeBehind.settings.RTT);
        component.Properties["Track_Boom"].Value = ("Boom_1500");
        component.Properties["Track_Pos"].Value = new Vector3(0, 0, 0);

        //Modify Fence Length and Position
        if (settings.HSTS_capacity == "3000Kg" | settings.HSTS_capacity == "5000Kg")
        {
            component.Properties["Fence_Length1"].Value = Math.Ceiling((2000 +
CodeBehind.settings.RTT) * 3.28084 * .001);
            component.Properties["Fence_Length2"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4000) * .001));
            component.Properties["Fence_Length3"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4000) * .001));

            component.Properties["Fence_Pos1"].Value = (new Vector3(((-2063.5) * .001), ((-2114.3) *
.001), 0));
            component.Properties["Fence_Pos2"].Value = (new Vector3((-2000 * .001), (-2000 * .001), 0));
            component.Properties["Fence_Pos3"].Value = (new Vector3((-2000 * .001), (Math.Ceiling((2000
+ CodeBehind.settings.RTT) * 3.28084 * .001)*309.48 - 1969.28) *.001, 0));

            CodeBehind.settings.Curtain_Transform_Y = -2095.887 * .001;
            CodeBehind.settings.Curtain_Transform_X = (Math.Ceiling(3.28084 * .001 *
Convert.ToInt32(CodeBehind.settings.HSTS_riser_height)) * 309.48 + 1961.08) * .001;
            CodeBehind.settings.Curtain_Len = (Math.Ceiling((2000 + CodeBehind.settings.RTT) * 3.28084 *
.001) * 309.48 - 2160.69);
        }

        if (settings.HSTS_capacity == "20000Kg")
        {
            component.Properties["Fence_Length1"].Value = Math.Ceiling((4000 +
CodeBehind.settings.RTT) * 3.28084 * .001);
            component.Properties["Fence_Length2"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4000) * .001));
            component.Properties["Fence_Length3"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4000) * .001));

            component.Properties["Fence_Pos1"].Value = (new Vector3(((-2000 - 63.5) * .001), ((-3000 -
114.3) * .001), 0));
            component.Properties["Fence_Pos2"].Value = (new Vector3((-2000 * .001), (-3000 * .001), 0));
            component.Properties["Fence_Pos3"].Value = (new Vector3((-2000 * .001), (Math.Ceiling((4000
+ CodeBehind.settings.RTT) * 3.28084 * .001) * 309.48 - -2951.24) * .001, 0));

            CodeBehind.settings.Curtain_Transform_Y = -3095.887 * .001;
```

```
            CodeBehind.settings.Curtain_Transform_X = (Math.Ceiling(3.28084 * .001 *
Convert.ToInt32(CodeBehind.settings.HSTS_riser_height)) * 309.48 + 1961.08 + 295.44) * .001;
            CodeBehind.settings.Curtain_Len = (Math.Ceiling((4000 + CodeBehind.settings.RTT) * 3.28084 *
.001) * 309.48 - 2160.69);
        }

        if (settings.HSTS_capacity == "30000Kg")
        {
            component.Properties["Fence_Length1"].Value = Math.Ceiling((4000 +
CodeBehind.settings.RTT) * 3.28084 * .001);
            component.Properties["Fence_Length2"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4000) * .001));
            component.Properties["Fence_Length3"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4000) * .001));

            component.Properties["Fence_Pos1"].Value = (new Vector3(((-2000 - 63.5) * .001), ((-3000 -
114.3) * .001), 0));
            component.Properties["Fence_Pos2"].Value = (new Vector3((-2000 * .001), (-3000 * .001), 0));
            component.Properties["Fence_Pos3"].Value = (new Vector3((-2000 * .001), (Math.Ceiling((4000
+ CodeBehind.settings.RTT) * 3.28084 * .001) * 309.48 - 1969.28 - 981.96) * .001, 0));

            CodeBehind.settings.Curtain_Transform_Y = -3095.887 * .001;
            CodeBehind.settings.Curtain_Transform_X = (Math.Ceiling(3.28084 * .001 *
Convert.ToInt32(CodeBehind.settings.HSTS_riser_height)) * 309.48 + 1961.08 +295.44) * .001;
            CodeBehind.settings.Curtain_Len = (Math.Ceiling((4000 + CodeBehind.settings.RTT) * 3.28084 *
.001) * 309.48 - 2160.69);
        }

        if (settings.HSTS_capacity == "50000Kg")
        {
            component.Properties["Fence_Length1"].Value = Math.Ceiling((5000 +
CodeBehind.settings.RTT) * 3.28084 * .001);
            component.Properties["Fence_Length2"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4500) * .001));
            component.Properties["Fence_Length3"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4500) * .001));

            component.Properties["Fence_Pos1"].Value = (new Vector3(((-2000 - 63.5) * .001), ((-3500 -
114.3) * .001), 0));
            component.Properties["Fence_Pos2"].Value = (new Vector3((-2000 * .001), (-3500 * .001), 0));
            component.Properties["Fence_Pos3"].Value = (new Vector3((-2000 * .001), (Math.Ceiling((5000
+ CodeBehind.settings.RTT) * 3.28084 * .001) * 309.48 - 1969.28 - 1496) * .001, 0));

            CodeBehind.settings.Curtain_Transform_Y = -3595.887 * .001;
            CodeBehind.settings.Curtain_Transform_X = (Math.Ceiling(3.28084 * .001 *
Convert.ToInt32(CodeBehind.settings.HSTS_riser_height)) * 309.48 + 1961.08 + 137 + 500) * .001;
            CodeBehind.settings.Curtain_Len = (Math.Ceiling((5000 + CodeBehind.settings.RTT) * 3.28084 *
.001) * 309.48 - 2160.69);
```

```
        }

        if (settings.HSTS_capacity == "70000Kg")
        {
            component.Properties["Fence_Length1"].Value = Math.Ceiling((5000 +
CodeBehind.settings.RTT) * 3.28084 * .001);
            component.Properties["Fence_Length2"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4500) * .001));
            component.Properties["Fence_Length3"].Value = Math.Ceiling(3.28084 *
((Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) + 4500) * .001));

            component.Properties["Fence_Pos1"].Value = (new Vector3(((-2000 - 63.5) * .001), ((-3500 -
114.3) * .001), 0));
            component.Properties["Fence_Pos2"].Value = (new Vector3((-2000 * .001), (-3500 * .001), 0));
            component.Properties["Fence_Pos3"].Value = (new Vector3((-2000 * .001), (Math.Ceiling((5000
+ CodeBehind.settings.RTT) * 3.28084 * .001) * 309.48 - 1969.28 -1496) * .001, 0));

            CodeBehind.settings.Curtain_Transform_Y = -3595.887 * .001;
            CodeBehind.settings.Curtain_Transform_X = (Math.Ceiling(3.28084 * .001 *
Convert.ToInt32(CodeBehind.settings.HSTS_riser_height)) * 309.48 + 1961.08 +137 +500) * .001;
            CodeBehind.settings.Curtain_Len = (Math.Ceiling((5000 + CodeBehind.settings.RTT) * 3.28084 *
.001) * 309.48 - 2160.69);
        }

        //Set Light Curtain Position
        System.Text.StringBuilder path = new System.Text.StringBuilder("S:\\Library
Build\\Custom_Lib\\Light_Curtain\\WolfLightCurtain_");
        System.Text.StringBuilder sb = new System.Text.StringBuilder("Curtain_");
        System.Text.StringBuilder sb_light = new System.Text.StringBuilder("Light_");

        if (Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) < 2500)
        {
            path.Append("8ft.rslib");
            sb.Append("8ft_0");
            sb_light.Append("8ft_0");
        }

        if (Convert.ToInt32(CodeBehind.settings.HSTS_riser_height) >= 2500)
        {
            path.Append("12ft.rslib");
            sb.Append("12ft_0");
            sb_light.Append("12ft_0");
        }

        GraphicComponent Light_check;
        station.GraphicComponents.TryGetGraphicComponent(sb.ToString(), out Light_check);
        //GraphicComponentLibrary mylib;
        mylib = GraphicComponentLibrary.Load(path.ToString(), true);
```

```
        Mechanism Curtain = (Mechanism)mylib.RootComponent.CopyInstance();
        Curtain.Name = sb.ToString();
        station.GraphicComponents.Add(Curtain);
        Curtain.DisconnectFromLibrary();

        double[] join_val = new double[] { (CodeBehind.settings.Curtain_Len + 2046) * .001 };
        Curtain.SetJointValues(join_val, false);

        //Get Current Joint Values
        double[] CurtainLength;
        CurtainLength = Curtain.GetJointValues();
        Curtain.Frames.ToArray();

        //Create Light Curtain
        Part Light = new Part();
        Light.Name = sb_light.ToString();
        station.GraphicComponents.Add(Light);

        //Move Light Curtain to Correct Location
        Matrix4 rot = Curtain.Transform.GlobalMatrix;
        Vector3 size = new Vector3(CurtainLength[0] + .13096, .03048, 1.20371);

        //Create Body and set Color and Transparency
        Body square = Body.CreateSolidBox(rot, size);
        square.Transform.Z = .33439;
        square.Name = "box";
        Light.Bodies.Add(square);
        VSTABridge.SetColor(Light, 255, 0, 0, 126);

        //Attach box to base of mechanism
        GraphicComponent home;
        Curtain.GetParentLink(0, out home);
        Part home1 = (Part)home;
        home1.Attach(Light, false, new Matrix4(new Vector3(0, 0, 0)));

        //Place Cutrain
        Curtain.Transform.X = (CodeBehind.settings.Curtain_Transform_X);
        Curtain.Transform.Y = (CodeBehind.settings.Curtain_Transform_Y);
        Curtain.Transform.RZ = 90 * (3.14 / 180);

    }
  }
}
```

XML

```
<?xml version="1.0" encoding="utf-8" ?>
<lc:LibraryCompiler xmlns:lc="urn:abb-robotics-robotstudio-librarycompiler"
                                                                    xmlns="urn:abb-
robotics-robotstudio-graphiccomponent">
 <lc:Library fileName="Auto_Creator_Full.rslib">
  <lc:DocumentProperties>
   <lc:Author>Joe</lc:Author>
   <lc:Image source="Auto_Creator_Full.png"/>
  </lc:DocumentProperties>
  <SmartComponent name="Auto_Creator_Full" icon="Auto_Creator_Full.png"

        codeBehind="Auto_Creator_Full.CodeBehind,Auto Creator Full.dll"

        canBeSimulated="false">

   <!--Inputs xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx-->
   <Properties>
    <DynamicProperty name="Weight" valueType="System.Int32" value="0">
     <Attribute key="MinValue" value="0"/>
    </DynamicProperty>
    <DynamicProperty name="Diameter" valueType="System.Int32" value="0">
     <Attribute key="MinValue" value="0"/>
    </DynamicProperty>
    <DynamicProperty name="Length" valueType="System.Int32" value="0">
     <Attribute key="MinValue" value="0"/>
    </DynamicProperty>

   <!--Outputs xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx-->
   <DynamicProperty name="Fence_Length1" valueType="System.Int32" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
   <DynamicProperty name="Fence_Length2" valueType="System.Int32" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Fence_Length3" valueType="System.Int32" value="0" readOnly="true">
     <Attribute key="MinValue" value="0"/>
     <Attribute key="Quantity" value="None"/>
    </DynamicProperty>
    <DynamicProperty name="Fence_Pos1" valueType=" ABB.Robotics.Math.Vector3" value="0,0,0"
readOnly="true">
     <Attribute key="Quantity" value=" Length "/>
    </DynamicProperty>
    <DynamicProperty name="Fence_Pos2" valueType=" ABB.Robotics.Math.Vector3" value="0,0,0"
readOnly="true">
```

```
      <Attribute key="Quantity" value=" Length "/>
     </DynamicProperty>
     <DynamicProperty name="Fence_Pos3" valueType=" ABB.Robotics.Math.Vector3" value="0,0,0"
readOnly="true">
      <Attribute key="Quantity" value="Length"/>
     </DynamicProperty>

    <DynamicProperty name="Track_Length" valueType="System.Int32" value="0" readOnly="true">
      <Attribute key="MinValue" value="0"/>
      <Attribute key="Quantity" value="None"/>
     </DynamicProperty>
    <DynamicProperty name="Track_Gantry" valueType="System.String" readOnly="true">
     </DynamicProperty>
    <DynamicProperty name="Track_Boom" valueType="System.String" readOnly="true">
     </DynamicProperty>
     <DynamicProperty name="Track_Pos" valueType="ABB.Robotics.Math.Vector3" value="0,0,0"
readOnly="true">
      <Attribute key="Quantity" value="Length"/>
     </DynamicProperty>

     <DynamicProperty name="Fence_Height" valueType="System.Int32" value="0" readOnly="true">
     </DynamicProperty>

     </Properties>
    <Bindings>
    </Bindings>
    <GraphicComponents>
    </GraphicComponents>
    <Signals>
     <IOSignal name="Execute" signalType="DigitalInput"/>
    </Signals>
    <Assets>
     <Asset source="Auto Creator Full.dll"/>
    </Assets>
   </SmartComponent>
  </lc:Library>
</lc:LibraryCompiler>
```