Benefit of Converting to RSLogix 5000 from RSLogix 500

Richard P. Crum, Jayme L. Davis, and Dr. Peter J. Shull
The Pennsylvania State University, Altoona Campus

Abstract

In conjunction with Creative Pultrusions, Inc., a fiberglass reinforced polymer composites manufacturer in Alum Bank, PA, a senior project was designed to convert the machine operation code for their pultruders from the Rockwell Automation's RSLogix500 software to the RSLogix5000 software. This project was a capstone design for the Electro-Mechanical Engineering Technology program at Penn State Altoona. The specific aim was to show the benefits of RSLogix5000 while improving the pultruding system at Creative Pultrusions, Inc. By streamlining the existing code, troubleshooting could become more efficient. In order to convert the code, a complete understanding of the pultrusion process was necessary along with that of both the RSLogix500 software and RSLogix5000 software. This document will discuss background information pertaining to Creative Pultrusions, Inc., RSLogix500 software and RSLogix5000 software in addition to the machine code conversion and testing processes.

Introduction

In the vast world of automated manufacturing, programmable logic controllers (PLCs) are one of the most reliable and effective means of controlling any electro-mechanical process. Creative Pultrusions, Inc. is a company, which implements the use of PLCs throughout their manufacturing process. Creative Pultrusions, Inc., a high strength pultruded fiberglass reinforced polymer composites manufacturer, was the foundation of the project research. The project was to convert the existing program, which is in RSLogix500, to the latest Rockwell PLC software, the RSLogix5000.

Pultrusion Process

Pultrusion is one of the many processes of producing fiberglass reinforced polymer composites. The pultrusion process starts by feeding fiberglass, woven fabrics, continuous strand mat, or carbon material through a series of creels[1]. This aligns and guides the material for entry into the die. As the material enters the die, it is impregnated with resin. Polyester, vinyl ester, and epoxies are typical resins that are used[1]. Excess resin is recycled back through the process. Once in the die, the heating process begins. The die has multiple heat zones to cure the material. Depending on the product, the set point for these zones will vary. The material is cured by an exothermic chemical reaction. There are catalysts in the resin that react once a certain temperature is reached. Heat will continue to be released by this process and allow the internal temperature of the product to exceed that of the die walls. After curing, the product is extracted from the die. The material is pulled through the pultruding machine by two hydraulic clamps. An encoder is used to monitor the length of material pultruded. When a desired length has been pulled through the machine, a saw cuts the product to length and resets the encoder. The saw can also be activated manually. Manual activation will reset the encoder back to zero. In order for continuation of the process, a flying cutoff saw is used. The product can then be unloaded.

RSLogix500

The pultrusion machine is controlled by the use of PLCs. The SLC 505 platform utilizing RSLogix500 software is used in the current machines at Creative Pultrusions, Inc. The RSLogix500 uses basic ladder logic to control a process. The commands used are set up through data files. The data files are arranged by memory storage with specific names/numbers for each bit. The language is easy to use for those who know it. The learning curve is also relatively easy. However the programs created can be lengthy. Simple logic is often used such as comparison statements, timers, and move commands. There are more complicated actions to be used in RSLogix500. Up to this point, the project research has turned up little use of these. The number span is limited depending on the command used, due to the number of bits allowable. For example a counter can only count up to 9999. Therefore, multiple counters need to be used for applications that require a count higher than this. Other commands are limited to ± 32,767 for integers. For scaling purposes, this could limit the tolerance on some processes. However, the float values offer a much larger range of numbers. RSLogix500 has been implemented with success for many years at Creative Pultrusions, Inc. There has been a common problem though: troubleshooting any dilemma is a complicated task. People who are knowledgeable in the RSLogix500 programming may have difficulty deciphering the code. This could be due to the fact that several programmers have made changes to this code over the years; each one possessing a different programming style. With well over 900 rungs of ladder logic and 200 pages of code, finding a problem can be cumbersome. Even though there has been success with the SLC505 and RSLogix500, improvements can still be made to the control of the machine.

Machine Control

There are several areas of interest with respect to the operational control of the pultrusion machine that were analyzed in this project. As far as the PLC is concerned, these areas of main interest are broken down as follows: heat zones, puller, saw, and hydraulics. Hydraulics is a topic that will be interwoven throughout the machine functionality. Currently, the Allen-Bradley SLC505 is being used in conjunction with Rockwell Automation's RSLogix500 programming software and RSView for HMI Display.

The main control of the heat zones in RSLogix500 is through PID (Proportional, Integral, Derivative) control. The set point, the desired temperature, for this system is initiated through the HMI. Depending on the product being pultruded, the set point will vary. Feedback needed for the PID operation comes from a thermocouple input. This is entered into the process variable, the current temperature, for the system. The controller output, CVEU, then determines the pulse sequence that will be implemented through the silicon-controlled rectifier (SCR). The SCR is used for pulse width modulation (PWM). The SCR pulses power to the heating elements in the product die. The PWM is currently controlled through the use of timers. The timers will set the pulse period depending on the controller output from the PID.

Two cooling systems have been implemented, one at both the entrance and the exit of the die. These are controlled by simple open/close valve control that depends on the temperature of the cooling zone. In the current ladder logic, simple comparison statements are used. Because of the simplicity, this portion is lengthier than it need be. Cooling has been set to have a tolerance of ±3ºF. If the temperature goes above the upper limit, the valve opens. If the temperature goes below the lower limit, the valve closes. The cooling portion of the die is not needed for all products.

With the web of control that is used for the heat zones, alarms are a must. Fault alarms sense if there are any connection problems, over or under heating situations, or disallowable user inputs from the HMI. The alarms appear on an alarm screen on the HMI, accompanied by the sounding of an audible alarm and flashing light. In order for the process to resume, the alarms must be cleared and reset. There are also certain conditions that must exist in order for the heating to initialize. These conditions can vary, but a few examples are: a push button on or no alarms activated.

The most difficult routine in this system is by far the pulling arrangement. An extremely complex program that is half the length of the total code controls the puller system. This complexity is partially due to the five different styles of pulling that can be implemented. Portions of the programmed code are repetitive. For example, the puller speed utilizes the same programming approach only differing by the style of pulling. Hydraulic cylinders power the pullers. A hydraulics card, available for the SLC 505, is used to control many of the settings for the hydraulic operations. This card uses two axes to control motion of the pullers. The third axis is used in the saw control, which will be discussed later. This card controls the speed, drift, acceleration, and deceleration of the pullers. Puller accuracy is determined by the tuning software and the quality of the servo valves. The pullers can be programmed to have purge modes. These modes stop the pulling cycle and help in keeping the die clear of debris. This task is very important for control purposes. Most of the puller code is implemented by using simple commands such as latches, moves, comparison statements, timers, and counters. These deal greatly with the position and force of the pullers. The hydraulics card plays a huge roll in this positioning; it keeps track of the exact location of the pullers at all times through feedback from the linear transducers and servo valving.

The cutoff saw is the last major step of the pultrusion machine. The saw movement is accomplished utilizing hydraulics and pneumatics. Vertical movement is performed through hydraulic cylinders. This is a discrete output to the hydraulic directional valve. The clamping and forward/reverse movements are done through pneumatics. The saw itself is a hydraulic controlled proportional valve. The PLC operates the saw through a sequential ladder logic program. The activation of the saw is performed through an encoder. The encoder monitors the amount of product pultruded. As with the other sections of this code, great care has been taken in setting up alarms. There is alarm code for any part of this process that could fault. This is possibly the easiest area to troubleshoot only because it has recently been rewritten. However, it is still a tedious job with over 70 rungs of ladder logic.

RSLogix5000

RSLogix5000 is the latest in the Allen-Bradley series of PLC software. RSLogix5000 uses one software package consisting of four styles of programming languages: ladder logic, structured text, function block diagrams, and sequential function charts. These programming languages can be used for control process, drives, sequential, and motion control[1]. This system lets the user create command labels through a tag-based platform. This allows the user to use a description of his/her choice for that command. The parameters for that command can then be specified to be a bit, integer, etc. Rockwell Software has stated the following about the RSLogix5000 platform[2]:

- Intuitive and simple to use
- Compliant IEC1131-3 interface
- Structured programming by way of symbols and arrays
- Instruction set supplying multiple applications
- Integrates DCS systems or single-loop controllers and dedicated servo or drive systems into one environment
- Online troubleshooting capabilities
- Ability to create new tags while online

Other capabilities are included but are beyond the scope of this document.

Code Conversion Process

It was decided that to fully assess the benefits and/or downfalls of applying RSLogix5000 to a real-world manufacturing process, some, if not all, of the code must be transcribed from its RSLogix500 form into the RSLogix5000 format. This was done in two steps. The first step was to decide which part of the pultrusion machine code would demonstrate the greatest contrast between the two software platforms. And the second step was the actual implementation of the new programming language.

Learning the RSLogix5000 System

The first step took considerable thought due to the vast possibilities of RSLogix5000's four programming languages that are available. Each language is best suited, but not limited to, specific electro-mechanical functionality. For example, ladder logic, which was the sole programming option with the RSLogix500 software, is bested suited for[2]:

- Execution of continuous or parallel operations
- Binary operations
- Logical operations that are complex
- Processing of messages and other communications

The function block diagram programming language can be used for:

- Drive control and continuous processes
- Control loops
- Circuit flow calculations

The sequential function chart (SFC) programming language works well in the following situations:

- Managing multiple operations at a high level
- Processes that are batched
- Motion control
- Sequential machine operations

Finally, the structured text programming language can be used in combination with some of the other RSLogix5000 programming languages. Rockwell Automation states in the *Logix5000 Controllers Common Procedures Programming Manual* that it is best suited for:

- "Complex mathematical operations

- Specialized array or table loop processing
- ASCII string handling or protocol processing"

For the scope of this paper, the function block diagram and sequential function chart programming languages will be used to accomplish our comparison between the RSLogix500 and the RSLogix5000.

The first step in converting the code was to set up the RSLogix5000 platform. This included installing the controller and modules needed for this project. Upon opening the software, a continuous task, program, and ladder logic routine have already been set up. The names of these can be changed to suit the application. Only one continuous task may be used in the project. A continuous task is one that will operate on a continual basis as long as the project is running. However, many periodic tasks can be used. The periodic tasks need to be set up for a time period. For example, this task could run every 500 milliseconds. Each task will have a corresponding program. The program will contain any created routines. Each program must have a main routine specified that will execute first. Within the routines, tags can be added. These tags may be program scoped (only available for use within the given program) or controller scoped (available for use throughout the project).

Heating Zones

One of the main objectives of the project was to condense the RSLogix500-based code utilizing one of the available languages in RSLogix5000. The heat zone code proved to be a prime candidate for this conversion. It was decided that function block diagram programming would be very effective in this situation. The main reason for this decision was that the pultrusion process utilizes a PID command for temperature control. The RSLogix5000 provides a PID command in the ladder logic and the function block diagram programming languages only. It was decided that function block diagram programming could demonstrate the best approach for controlling the heating zones.
Also, the heat zones are a continuous process which Rockwell software recommends function block programming for this type of control.

The first obstacle was the implementation of the PIDE (Enhanced PID). The PIDE was created in a periodic task[2] due to suggestion from the *Logix5000 Controllers Process Control and Drives Instructions* manual from Allen-Bradley. All tags were created as controller tags, giving access to these tags from any routine. The most difficulty came from deciding which inputs and outputs to use. There are over 140 inputs and outputs to choose from. Some of these parameters are shown in figure 1. The PIDE can be used for a vast amount of control. In this project, basic PID control was sought. The input used for the process variable was to signify that of a thermocouple. The reading of the input was controlled through an RSView generated testing screen. The set point came from a randomly chosen number, in this case 350°F. The control variable output was to control the heat pulsing. Tags were created for the thermocouple and set point inputs. As will be shown later, the control variable has a direct connection to another function block so a tag was not created.

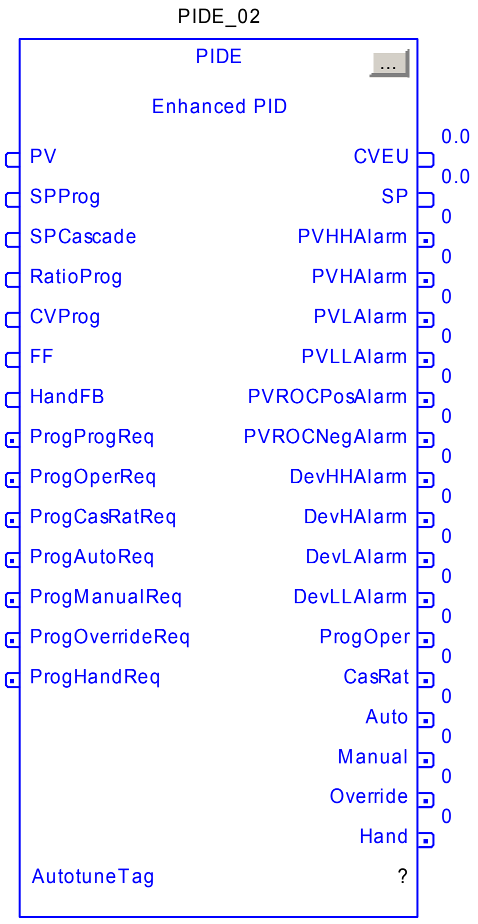


Figure 1 (PIDE Function Block)

Initial runs of the PIDE loop failed. Troubleshooting began with changes to the operational mode settings. There are many setting types, which can be used to control the PIDE. The major ones of concern to this project include auto or manual, program-auto, program-manual, operator-auto, operator-manual, program-program, program-operator, operator-program, operator-operator. Difficulty was had in determining the type of control needed to make the PIDE functional. It was important for the PIDE to run continuously without any user inputs. Therefore, program-auto request was used in conjunction with auto and program-program request. With this combination of controls the program will request the PIDE to run in automatic mode. There will be no operator control of this function.

After changing the above modes, no progress was made in running the PIDE loop. Through the online help menu of RSLogix5000, sample programs can be found. By comparing these programs with that of the project, certain differences were taken into consideration. One of these was the timing mode. The timing mode previously set was found in the RSLogix500 programming code. However in the conversion, it was found that a timing mode of 0, periodic mode, had to be used. This needs to be used due to the fact that the PIDE is in a period task. After these modes were changed, the PIDE was in working order. The values for P, I, and D were chosen to exemplify the functionality of the heat zone process loop. P, I, and D values will be adjusted according to individual processes.

The control variable output (CVEU) of the PIDE function block was used as an input to the heating element pulsing. The pulsing element used was a Split Range Time Proportional (SRTP) function block, shown in figure 2. This instruction converts the CVEU output from the PIDE into a digital pulse. This digital pulse output will be used to control the SCR. Problems were encountered during implementation of the SRTP. The SRTP is designed to control heating and cooling in one loop. The die for the pultrusion machine consisted of two separate loops governing the heating and cooling zones. An alternate method of heating element control was found utilizing the Position Proportional (POSP) function block. This instruction uses a cycle time based on the percentage of control variable output to pulse the SCR on and off.

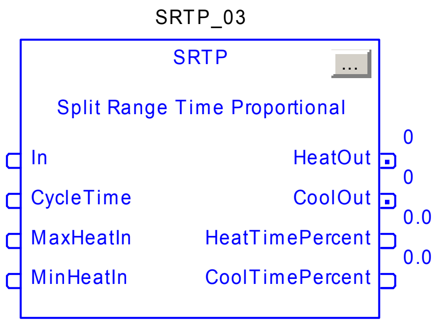


Figure 2 (Split Range Time Proportional)

In order for the POSP to control the SCR, shown in figure 3, the CVEU (PIDE) was wired to the set point input (POSP). The POSP has OpenOut and

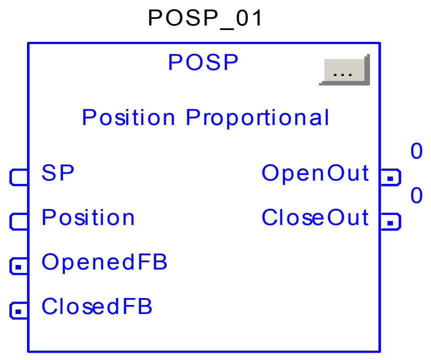


Figure 3 (Position Proportional)

CloseOut as outputs. Ideally, both of these outputs needed to be wired to the SCR, but the RSLogix5000 protocol would not permit this. The OpenOut was wired to the SCR input and appeared to be functioning correctly at first. After closer examination, it was found that the POSP output pulse was not proportional to the PIDE CVEU output. Further experimentation with the SRTP command demonstrated the possibility of effective operation within the heating zone code. The SRTP was made useful in the "heating only" code application by, in essence, eliminating the cooling output. This was obtained by entering a small value in the MinCoolIn parameter of the SRTP. With the newly programmed SRTP, the PID heating zone code appeared to be functioning correctly. Again, final adjustment of the SRTP parameters would be attained during actual application commissioning.

Some of the lower level programming involved in the heating zone routine consisted of the utilization of the Scale (SCL) command. This command is nothing new to Rockwell Software. It is found in both the ladder logic programming and the function block diagram programming formats. The SCL was used to scale the actual inputs from either the operator HMI or the thermocouples. The main idea being that the level of temperature control resolution needed to be maintained, if not exceeded, by the conversion from the RSLogix500 programming to the RSLogix5000 programming with respect to the die heating process.

With the above ideas several sheets were created for the heating zones. These sheets were identical except for the zone activated. It was found the length of code was cut down. This will be a benefit that will aid in troubleshooting if any problems were to occur. The conversion of the heating process code encompasses the major portions of the RSLogix500 code. The RSLogix5000 function block diagram does not cover every aspect of the original code. The complete programmed code is shown figure 4.

Cooling Zones

Initially, it was planned to include the heating and cooling control loops together in one central command FBD program routine. But, after more consulting with Creative Pultrusion's manufacturing engineers, it was found that these two processes needed to be controlled under very different circumstances. If the heating and cooling processes were aimed at attaining one temperature throughout, then the utilization of the SRTP heating and cooling output pulses would be very effective. This was not the case. The original RSLogix500 code consisted of a temperature low limit and a temperature high limit that are continuously compared to a thermocouple input. The conversion of the cooling zone was simple to implement Using nothing more than a Greater Than or Equal To (GEQ), a Less Than or Equal To (LEQ), and a Discrete 2-State Device (D2SD) command, the cooling valve was commanded full open or full closed. Refer to figure 5 for the D2SD command. It is worth noting that this process could be regulated

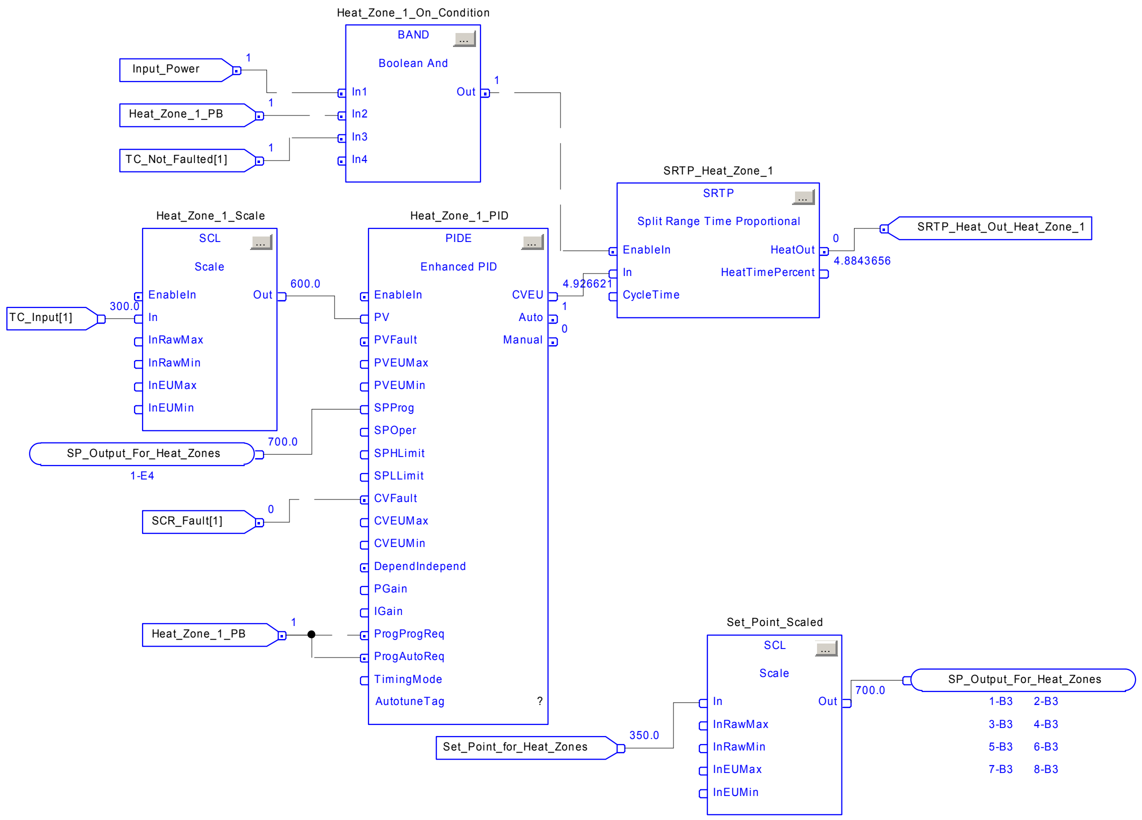


Figure 4 (New Heating Zone Code in Function Block Diagram as Programmed in RSLogix5000)

utilizing a PIDE function block loop, but the current pultrusion process does not warrant this magnitude of precision temperature control. The implementation of the D2SD was much simpler than trying to utilize a PIDE function block. To show the functionality of the D2SD an example will be given.

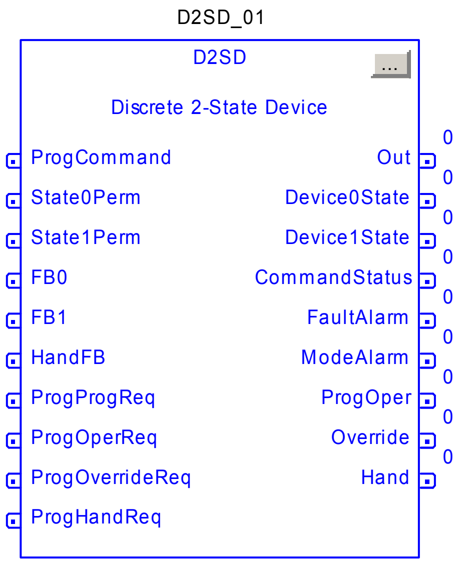


Figure 5 (Discrete 2-State Device)

A cooling zone has an ideal temperature of 40°F ±3°. The high limit in this situation is 43°F and the low limit is 37°F. A comparison will be made between the thermocouple input and the limits. As shown in figure 6, the high limit is connected to the ProgCommand; the low limit controls the State0Perm (State 0 Permissive). The D2SD looks at the ProgCommand to set the input high or low. If the temperature is 50°F, the valve will be open; Out will be high. If the temperature drops to 40°F, Out will still be a high. The State0Perm prohibits the output from going low until this input becomes high. Once the temperature drops below 37°F, Out will become a low. The cooling could also be set up in several other ways. For example, the State 1 Permissive (State1Perm) could be used to control the high limit.

The cooling zones are much easier to decipher with the conversion. In this case, the code was not condensed. However, it is felt that troubleshooting will be easier. The complete program for the cooling can be seen in figure 6.

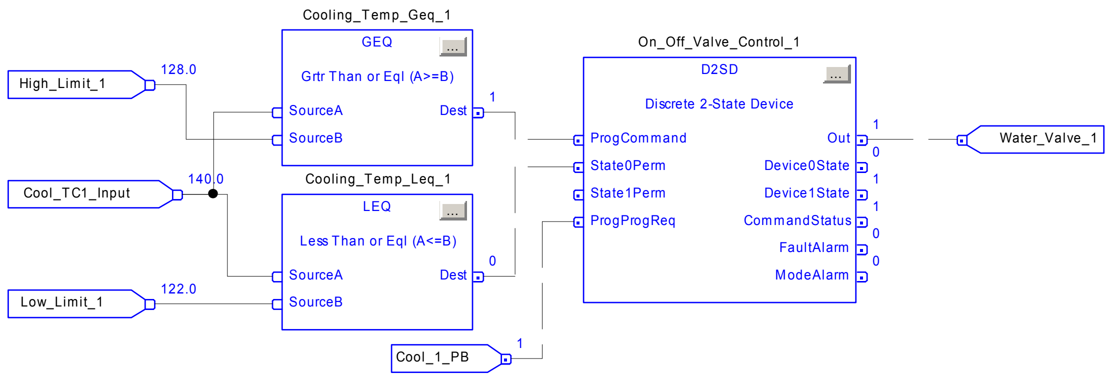


Figure 6 (New Cooling Zone Code in Function Block Diagram as Programmed in RSLogix5000)

Saw

Exploration into the realm of RSLogix5000's sequential function chart programming language was addressed utilizing Creative Pultrusion's flying cutoff saw code. After careful review of the existing RSLogix500-based program, it was found that the saw control could be accomplished with a handful of steps and transitions as compared to more than 70 rungs of ladder logic. An initial ladder routine was used to establish a starting point for the saw. Basically, once all start conditions are met, along with the encoder output reaching the HMI input desired part length, the program will jump to a SFC subroutine. This is where each step of the saw operation transpires. The SFC routine has the saw cycling through 5 steps, which each consist of one or more actions that are commanded. These 5 steps consist of the following operations: (1) part clamp down, blade motor on, dust gate on: (2) Saw head down, encoder output set to zero; (3) saw cross cut; (4) saw head up, blade motor off, dust gate off; (5) saw clamp up, saw to home position. Each step transitions by way of sensor inputs indicating the end of travel, or by the completion of an individual process in the overall makeup of the saw operation. This program can also be seen in figures 7 and 8.

Additional Program Upgrades

As time permitted throughout this project, different portions of the pultrusion machine programming were experimented with using the available languages contained within RSLogix5000. The basis for these experiments was to try to simplify not only the efficiency of operation, but also the opportunity for effective future troubleshooting of the code. One such area looked at was the Die Set-Up programming code. It was found that some repetitiveness was

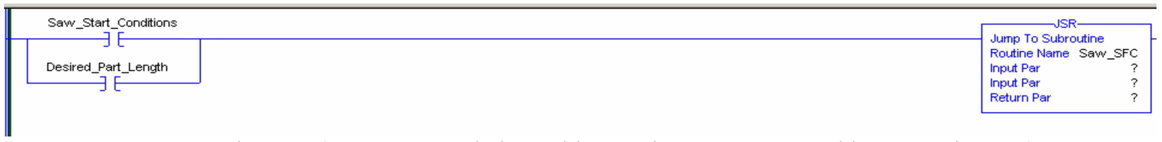
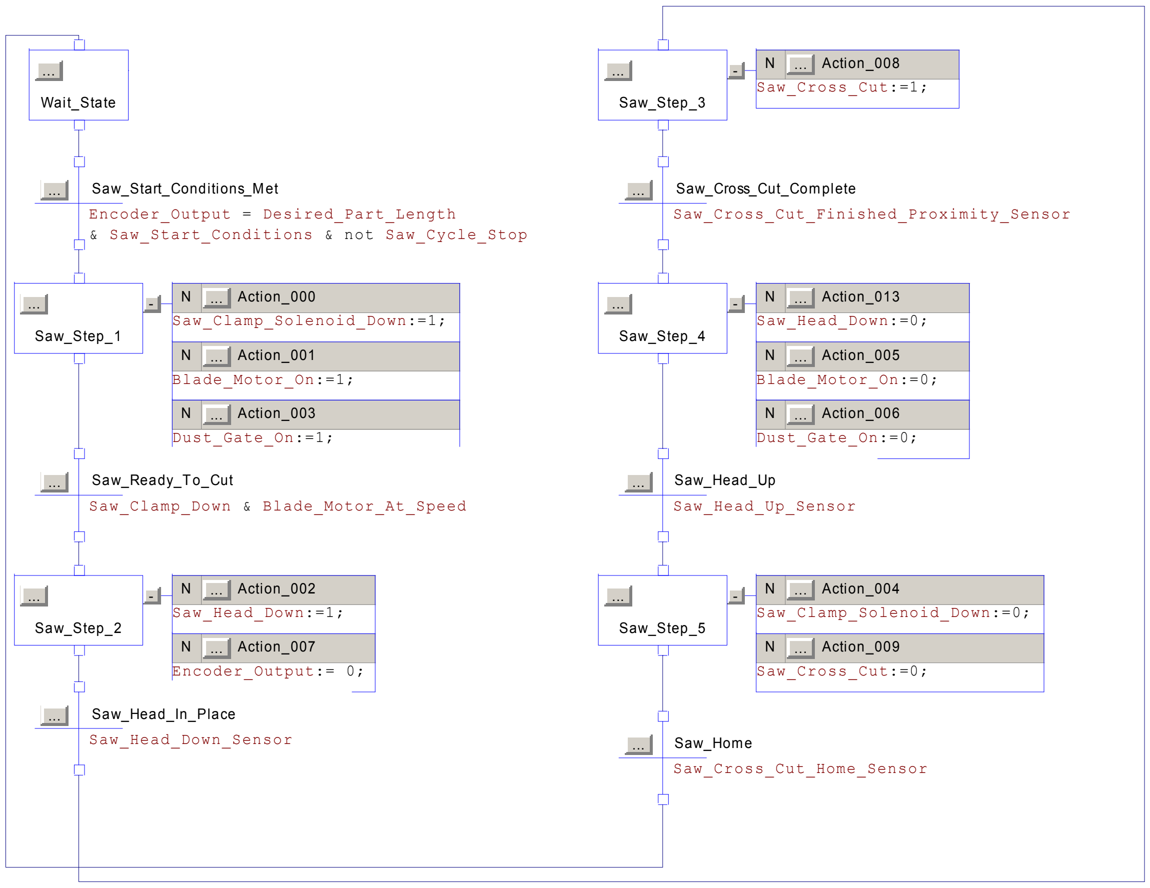


Figure 7 (New Saw Code in Ladder Logic as Programmed in RSLogix5000)



```
Encoder_Output = Desired_Part_Length
& Saw_Start_Conditions & not Saw_Cycle_Stop
Saw_Clamp_Solenoid_Down:=1;
Blade_Motor_On:=1;
Dust_Gate_On:=1;
Saw_Clamp_Down & Blade_Motor_At_Speed
Saw_Head_Down:=1;
Encoder_Output:= 0;
Saw_Head_Down_Sensor
Saw_Cross_Cut:=1;
Saw_Cross_Cut_Finished_Proximity_Sensor
Saw_Head_Down:=0;
Blade_Motor_On:=0;
Dust_Gate_On:=0;
Saw_Head_Up_Sensor
Saw_Clamp_Solenoid_Down:=0;
Saw_Cross_Cut:=0;
Saw_Cross_Cut_Home_Sensor
```

Figure 8 (New Saw Code in Sequential Function Chart as Programmed in RSLogix5000)

involved when looking at the prerequisites to obtain die table lift and lower output commands. The idea was to have one XIC input command the activation of the table up and the table down movements. This was done using a small FBD program. This routine can be seen in figures 8 and 9. This program consisted of all the potential prerequisites flowing through a Boolean Or (BOR) and a Boolean And (BAND) function block combination. This combination had a single output, which was given the same tag name as the XIC input in the ladder routine. With further research, it is conceivable that the whole Die Set-Up program could be accomplished with one simple FBD routine.

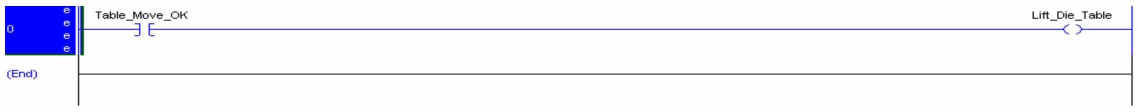


Figure 8 (New Die Set-Up Code in Ladder Logic as Programmed in RSLogix5000)

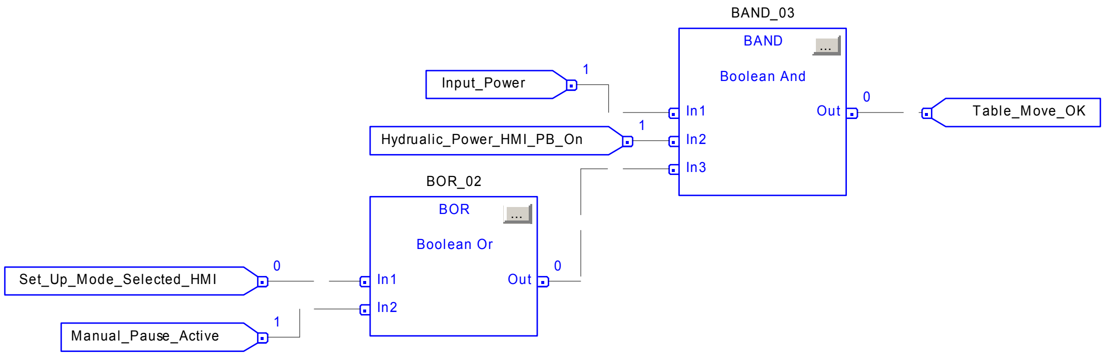


Figure 9 (New Die Set-Up Code in Function Block Diagram as Programmed in RSLogix5000)

Conclusion

The partial conversion of the pultrusion machine programming from RSLogix500 to RSLogix5000 has made clear that there are benefits to be attained. It can be seen that the programming versatility alone could make the conversion from the RSLogix500 platform to the RSLogix5000 platform a worthwhile venture. With four different programming languages at your disposal, almost every electro-mechanical control application conceivable can be accomplished through the implementation of Rockwell Software's RSLogix5000 software. Due to the multiple programming languages, the code was able to be shortened. Each programming language has a topic that it has been designed for. With this in mind, the ladder logic pertaining to the heating zones was more suitable for function block diagrams than ladder logic. The commands are more suitable to particular applications and, therefore, shortened the code.

Another benefit is having the ability to program all inputs and outputs using descriptive tags. It was obvious through the project code writing processes that a program with tags would make a program not only easier to follow, but easier to troubleshoot. Additionally, the ability to monitor and change tag values, while the program is online, has been a benefit. For example, gain values for a PIDE function block can be changed online to optimize your PID curve. The result of this tuning can be seen almost instantaneously and will not disrupt the other program functions.

This project has found one main disadvantage to the conversion. Creative Pultrusions, Inc. is an example where it might not be beneficial to convert to RSLogix5000. Their pultrusion machines function well with their present RSLogix500 platform. Therefore, the cost for the platform upgrade and the accompanied learning curve for RSLogix5000 are not justifiable at this time.

Acknowledgements

We would like to thank the following people at Creative Pultrusions, Inc. for their contributions to this senior project: Bob Trostle, John Plunkard, and Jason Blough. We would also like to thank Dave Mayewski of Rockwell Software for his insight pertaining to the code conversion, and John Sjolander, Lab Coordinator at Penn State Altoona, for continuous help throughout the project.

References

1. Creative Pultrusions Inc. www.pultrude.com. Retrieved from Internet at Penn State University, Altoona Campus. September 25, 2003.
2. Rockwell Software. "RSLogix 5000 Enterprise Series Software Technical Data." 2001 Rockwell International Corporation.
3. Allen-Bradley and Rockwell Automation. "Logix5000 Controllers Common Procedures" Programming Manual. August 2002.
4. Allen-Bradley and Rockwell Automation. "Logix5000 Controllers Process Control and Drives Instructions" Reference Manual. August 2002.