MSE 310

Introduction to Electromechanical Sensors and Actuators

DUNCAN ENSING *301147103*

SOHAIL SANGHA *301186636*

MATT HARRON *301211346*

December 1st 2014

Contents

Introduction

The purpose of this project was to setup a robust automation system using LabVIEW and a National Instruments Data Acquisition Device (DAQ), specifically model USB-6501. What we controlled through the DAQ was a Bytronic ICT3 (Industrial Control Technology) that was designed to simulate a typical industrial automation system. The ICT demonstrates component sorting, part assembly and inspection processes using various sensors and actuators. The general concept for our LabVIEW program is to sort aluminum pegs and plastic hoops that are placed in the system randomly, then assemble the hoop on top of the peg and finally determine whether the assembly is complete. If the assembly is correct, let it pass through and if not, eject the part or assembly.

Over the course of the project, students are expected to build an automated program using LabVIEW that can achieve all the goals listed below. Additionally, they are expected to gain an understanding of the automation and controls business, relevant challenges , and the hardware and software currently being used in the automation industry.

Goals

The tasks students must perform are:

- Develop a graphical user interface (GUI)
- Display the number of assemblies hoops and pegs handled by the system
- Reject incomplete assemblies
- Display real time sensor signal states
- Show the efficiency of the system
- Implement a stop\pause button
- Design a system recovery feature in case of a power loss

System Overview

There are 4 types of sensors and 3 types of actuators used in the ICT. They include inductive, capacitive, photoelectric and fiber-optic sensors, as well as linear solenoids, electric motors, and a rotary solenoid.

Sensors

Inductive

The inductive sensors are used to detect the presence of an aluminum peg. This sensor is a self-induction transducer that consists of a single coil and detects metallic objects via a changing high frequency electromagnetic field. One of the main challenges with this type of sensor is its non-linearity, meaning it has a high sensitivity for close distances and low sensitivity for far distances. As you can see in Figure 1 below, the inductive sensor has to be extremely close to the target object. This aspect of the sensor is even more important when dealing with non-ferrous materials like the aluminum pegs. This sensor is used for 2 main reasons in the ICT. First, to detect the presence of an aluminum peg; second, to confirm the operation of the linear solenoids in the sorting and reject areas.

Capacitive

The capacitive sensor is used to detect the presence of a plastic ring at the sensing station. This sensor detects the change in an electric field between two objects. If the capacitive sensor detects a component and the inductive sensor does not, the systems knows the object is a plastic ring once is has passed thought the fiber-optic sensor and is later rejected by the linear solenoid. This sensor can be seen in Figure 2.

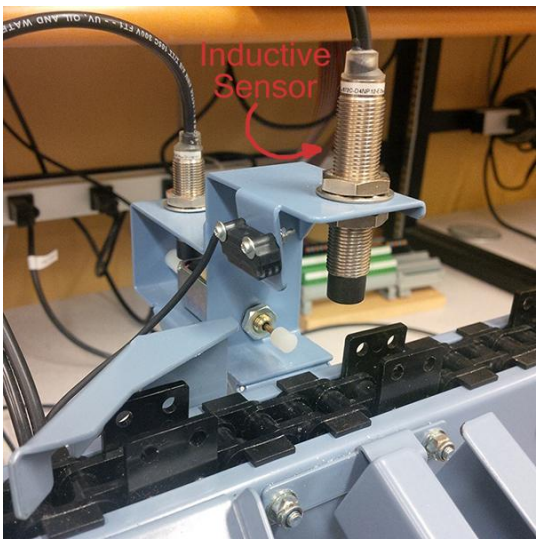

Figure 1 - Inductive and Photoelectric in the ICT sorting area

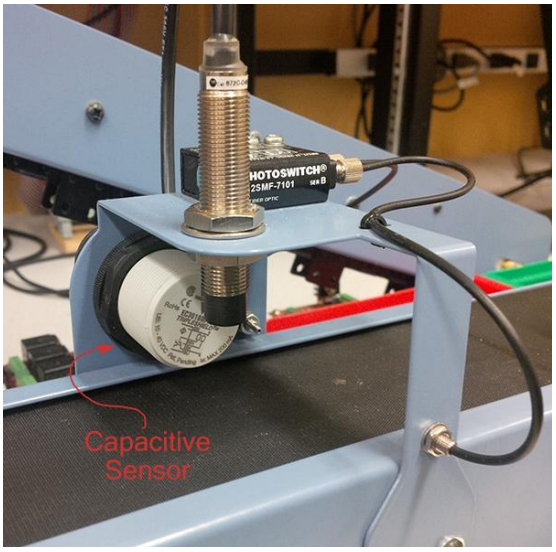

Figure 2 - Capacitive Sensor in the Sensing area of the ICT

Photoelectric

There are three spots on the ICT where the photoelectric sensors are used; the sorting area, assembly chute, and the reject area. These sensors are set up in a reflective type configuration, meaning the emitter and receiver are in one unit and require the light signal to bounce back from target object. This sensor is used as a double check in various stages of the ICT. For example, if the photoelectric switch detects an object without the inductive sensor detecting an item in the sorting area, the system can determine that there is a plastic ring in the assembly.

Fibre-optic

There is only one fibre-optic sensor used on the ICT, and it is used in the sensing station as shown in Figure 3. This sensors helps to determine if there is a part or an assembly going through the sensing area. This sensor works on the premise of a solid beam, and if that beam is disrupted, it then detects an object. These sensors are excellent to have on many systems as they are immune to electric, magnetic and light interference.

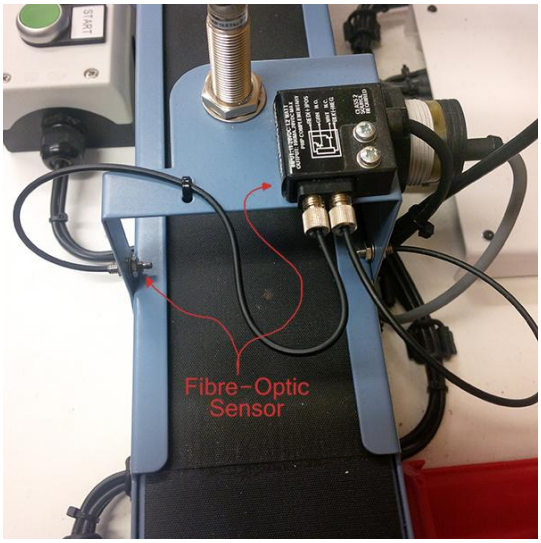

Figure 3 - Fibre-Optic Sensor in the Sensing Area of the ICT

Actuators

Linear Actuator

There are 2 linear actuators used in the ICT. One of the linear actuators is shown in Figure 4, and is part of the sensing area and the other is in the sorting area of the ICT. This linear

actuator pushes objects determined to be parts into the reject area/bin. These linear solenoids are used in closed loop operating in conjunction with the inductive sensor over the rear portion of the solenoid. The inductive sensor simply verifies the proper function of the actuator.

Rotary Actuator

There is one rotary actuator used in the ICT, located in the assembly chute seen in Figure 5. This rotary solenoid restricts how many plastic pegs are in the hopper to be assembled by a passing aluminum peg. Our program was set up to actuate the rotary solenoid when the photoelectric sensor did not detect a plastic ring in the hopper.

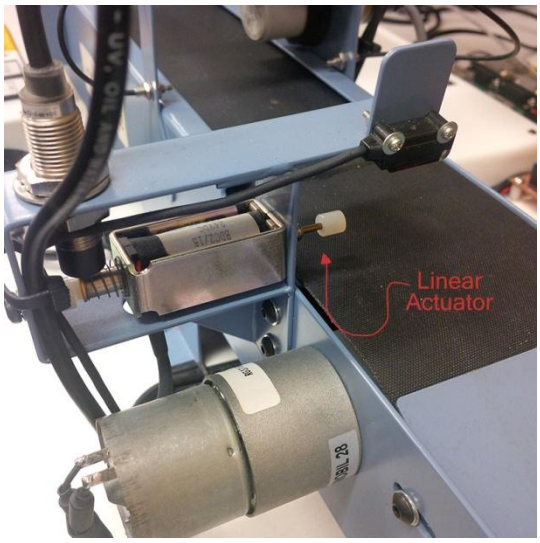

Figure 4 - Linear Actuator in the Sensing Area of the ICT

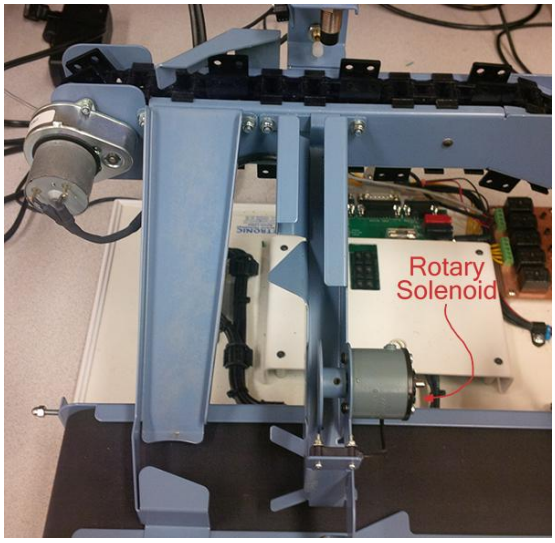

Figure 5 - Rotary Solenoid in the Assembly Chute of the ICT

Electric Motor

Electric motors are used extensively in automation and controls environments and this one is no different. There are two DC motors in the ICT. One of the motors drives the conveyor chain that feeds the sorting area on the top of the ICT with aluminum pegs and plastic rings. The other motor drives the belt on the lower side of the ICT, transferring parts and\or peg and ring assemblies to the sensing area before either being rejects or continuing on the assembled area and end of the ICT.

Algorithms & Flowcharts

Program Execution

Figure 6 below is a top-level representation of how our code executes. The program's first task after starting is to establish communication with the ICT system's inputs and outputs. As a safety precaution, all the outputs are immediately given an appropriate value to turn them off, regardless of their previous state.

The next step is to try recovering the program's memory from its previous run. All the necessary sensor and actuator values are stored in a text file which can be read out of before the program executes. One useful feature of the text recovery method used here is that users can manually enter their own data into the file to force outputs to a certain state or to account for unforeseen operating situations. This stage also presents the user with the option of resetting the system and starting fresh.

Once the memory recovery portion is complete, the main body of the program runs. This part is responsible for reading from sensors, writing to actuators, and logging the recovery data in a text file. It runs continuously until interrupted by the kill switch in the user interface or until the program is aborted. Pressing the stop button on the machine will only stop the belts but continue to poll the sensors.

Once the kill switch is pressed, the program writes to the recovery file a final time and exits the main loop. It finishes up by turning off all the actuators, closing Labview's links to the NIDAQ, and exiting the program. Refer to the appendix for more detailed flowcharts of each portion of our program.

There are a few parts of the program worth examining in greater detail; the sensor triggering, assembly tracking, and efficiency calculation. Labview is unable to use data from the NIDAQ in its raw state, so it must write the sensor value into a boolean variable first. This boolean is passed on to a shift register so that on the program's next iteration it can compare the sensor's previous value with its current value to detect rising or falling edges. Only once an edge is detected will the program execute the appropriate code.

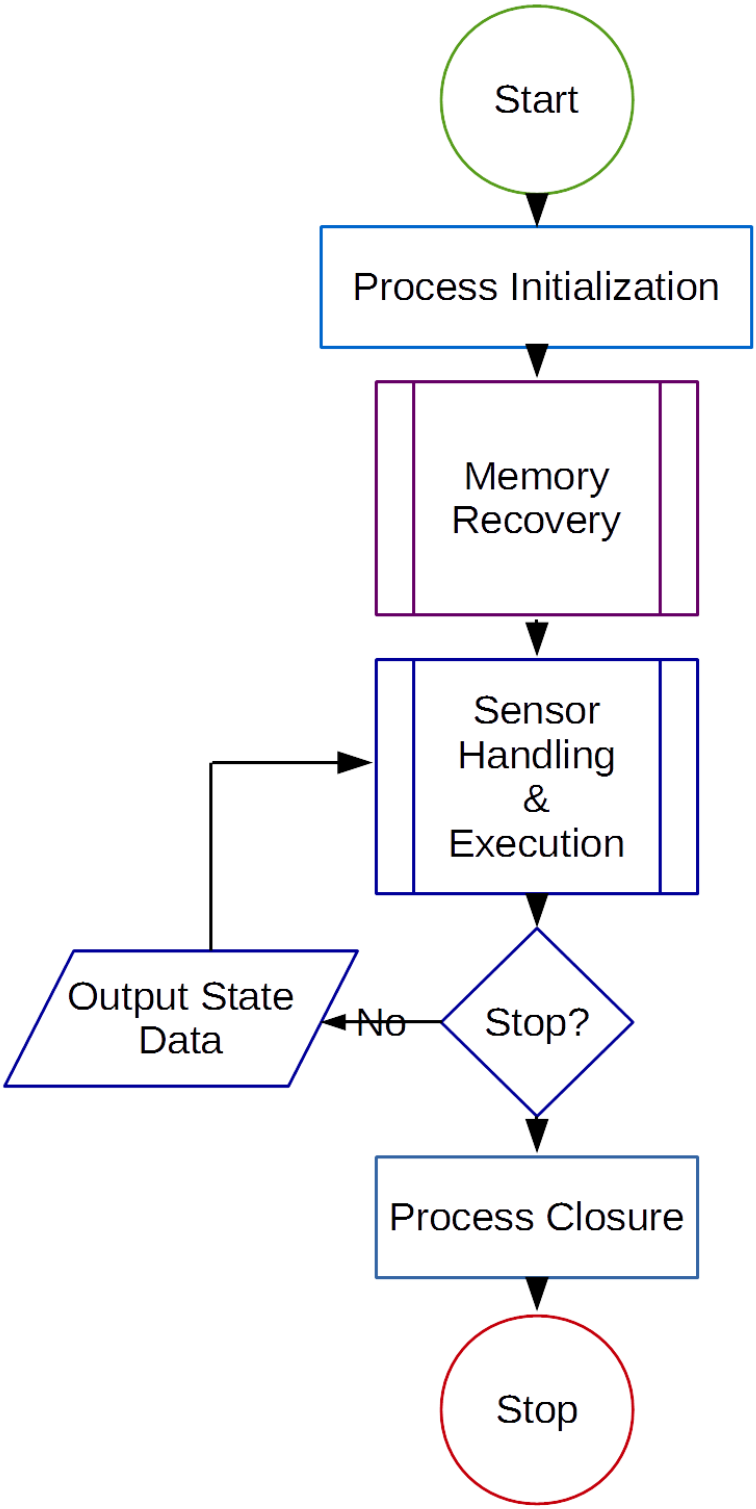

Figure 6 - Flowchart the code's top-level operation.

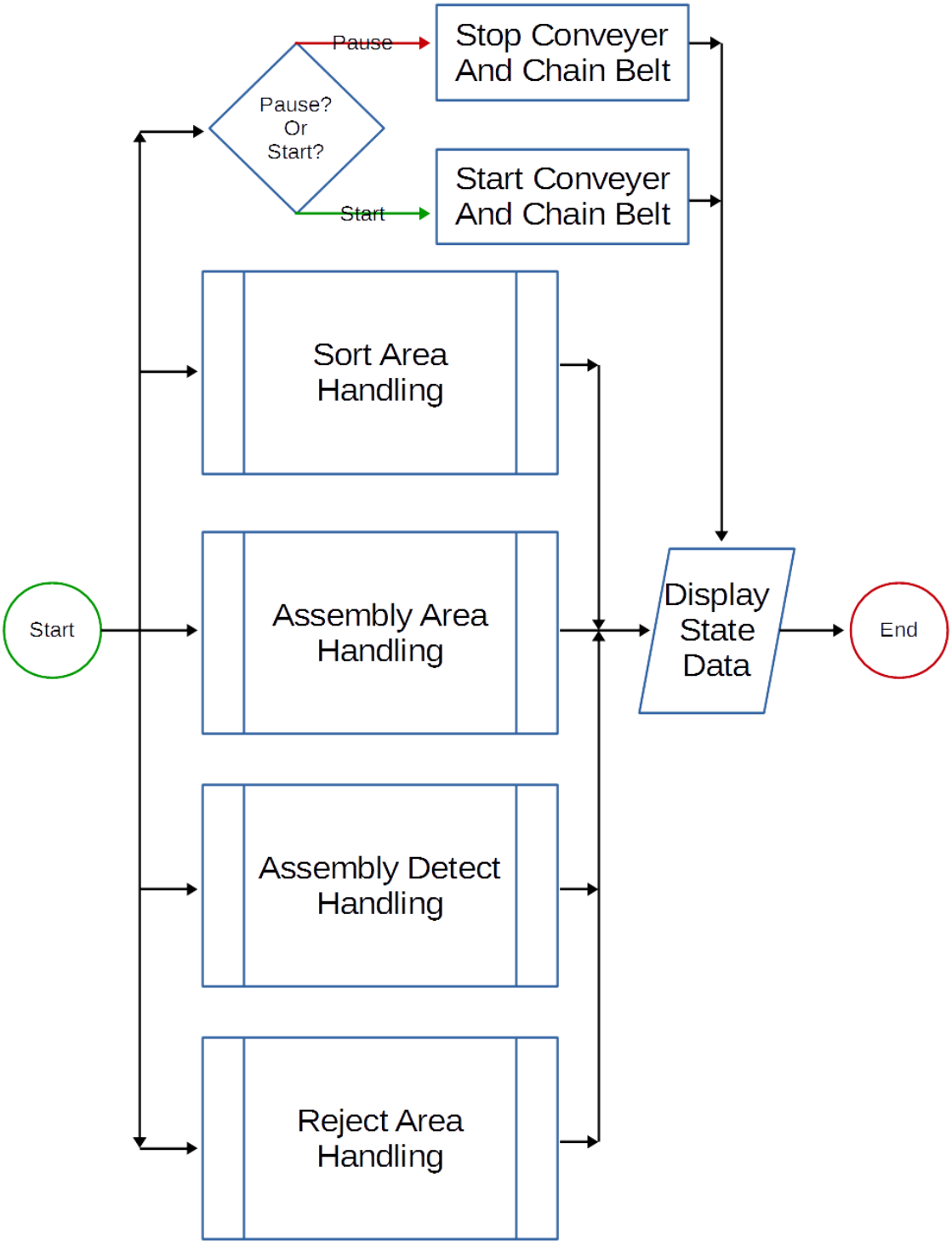

Figure 7 - Overall Sensor Handling

The program makes use of queues to track which widgets have been processed and how many assemblies have been created. Depending on the various sensor inputs, the program will write either a 1 or a 0 to the parts tracking queue to determine whether a hoop or a peg was processed. Another queue was established to track if the reject area was detecting a lone part or a complete assembly. Depending on the contents of these two queues, the program will choose to accept or reject the part and will modify the UI counters accordingly.

The efficiency calculation was straightforward with one small stumbling block. A full assembly consists of two widgets, so a factor of two must be inserted. The equation used was:

$$efficiency = \frac{2 * assemblies\ made}{widgets\ processed}$$

Troubleshooting

A number of bugs were encountered while writing the program for the ICT system. These were mostly simple to fix, such as actuators firing too early or late, but a few more significant issues presented themselves too. We will discuss only a few of the bugs we dealt with; selected to demonstrate a variety of problem types and their fixes.

The first issue was related to actuators firing at inappropriate times. The rotary solenoid in the hoop feeder is a good example of this behavior. It was originally set up such that it would open immediately once the first hoop was pushed into the feeder, but this caused the hoop to become stuck when the solenoid closed soon afterwards. To correct this problem, a software stopwatch was implemented. It is worth noting that we did not use a delay or wait function for the timer, which allows the main program to keep running while the timer counts in the background. The program was rewritten such that once the hoop was pushed into the feeder, the timer would give it enough time to settle in place before opening. The timer was also used to hold the solenoid open for a preset period before closing it again. A similar system was implemented for every solenoid in the system; a small, single iteration loop would contain all the code needed to activate, time, and deactivate the corresponding output.

The next problem was related to dealing with unwanted data from the sensors. This problem exposed itself most clearly at the top sorting area. Once the hoop feeder contained 5 hoops, any additional hoops would be allowed to pass by. Due to their donut shape, the

proximity sensor in the sorting area would be triggered twice, making the system think two widgets had passed by when there was only one. Software timers were used again to solve this problem. When the proximity sensor was triggered the first time, it would check a stopwatch that tracks the elapsed time since its last trigger. If the elapsed time was below a threshold it would ignore the input; if it was above the threshold it would accept the input and continue as normal. This feature was used on multiple sensors through the program to prevent false signals.

The final problem we will discuss is related to queues. This is not a problem that would ever be encountered in regular operation, but its consequences are serious enough that it should at least be addressed. Due to a mechanical problem, two assemblies would occasionally travel along the lower belt right next to each other. If they were close enough, the fibre optic sensor would only sense them as a single assembly and write a single element to the tracking queue. Then, when the assembles reach the rejection area the proximity sensor would still be able to detect two assemblies. This resulted in the program attempting to dequeue two elements from the tracking queue and causing a memory leak which completely locked up the program. We never ended up developing a solution to this issue because it was due to a hardware malfunction and was outside of the project's scope.

Summary & Conclusion

Overall, the ICT sensor system proved to be a useful and effective tool for teaching students how to implement event based programs in Labview. It provided valuable experience in both graphical programming and managing program flow. By the end of the project period we had achieved all that we set out to do, we: developed a robust and efficient program for assembling widgets, provided an effective means of program backup, and implemented a clutter free GUI which presented the end user with all the information they need.

Recommendations

Our only recommendation to improve the ICT is to raise the friction between the lower conveyor belt and pegs. Our biggest problem in the program was caused by pegs being too weak to pull a hoop out of the hopper and resulted in software issues and efficiency loss.

Appendix

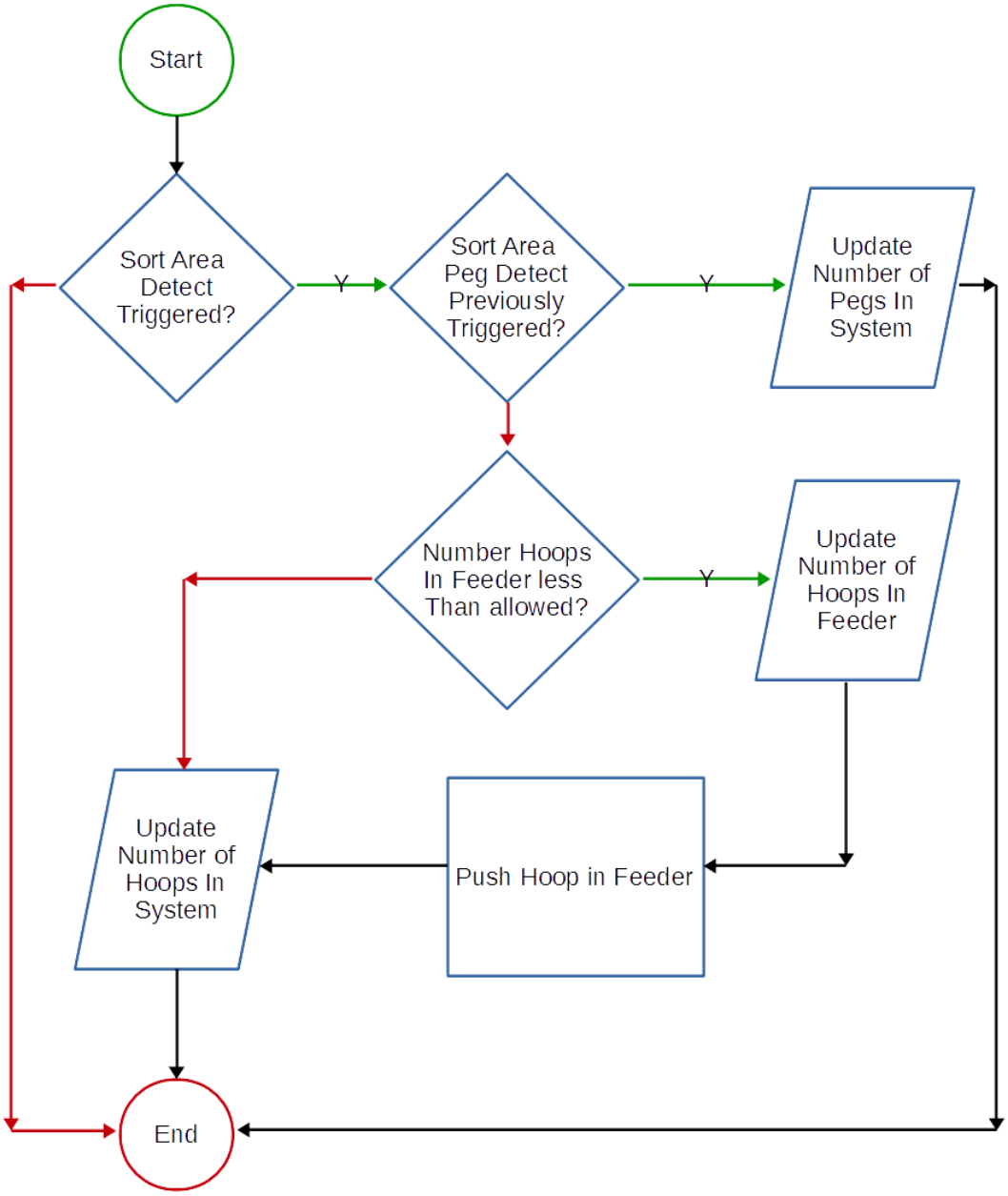

Figure 8 - Sort Area Handling

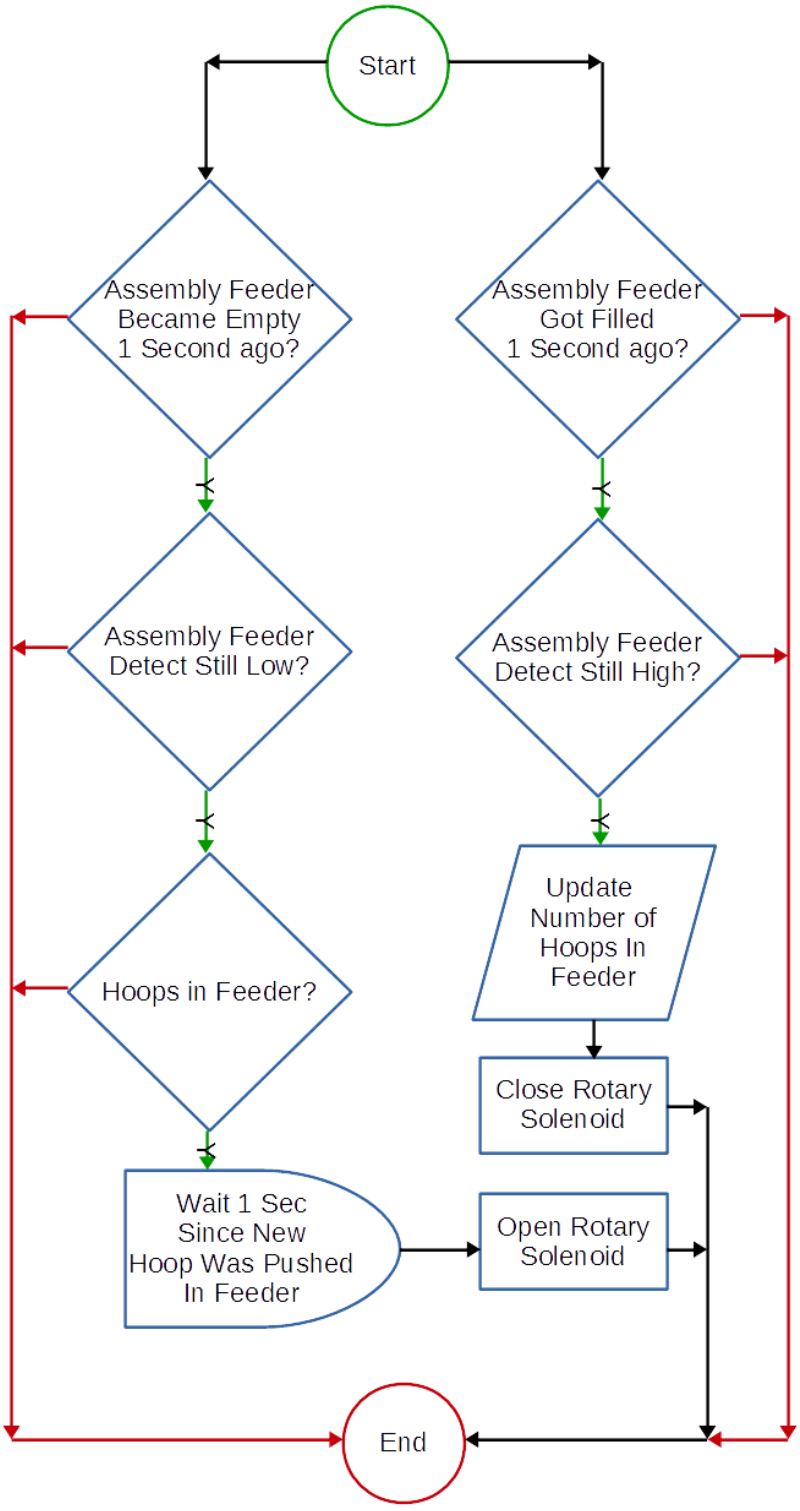

Figure 9 - Assembly Area Handling

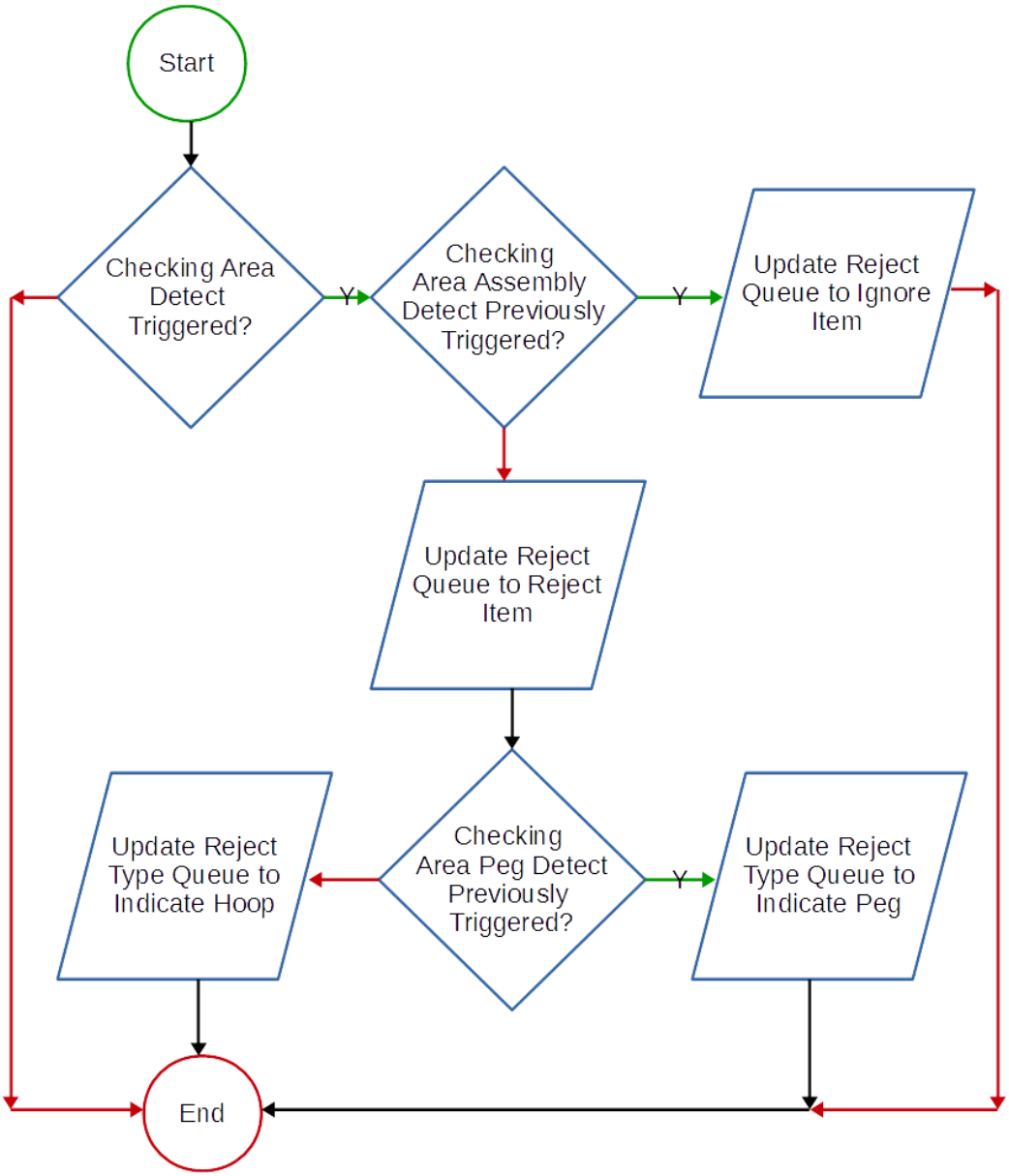

Figure 10 - Assembly Detect Handling

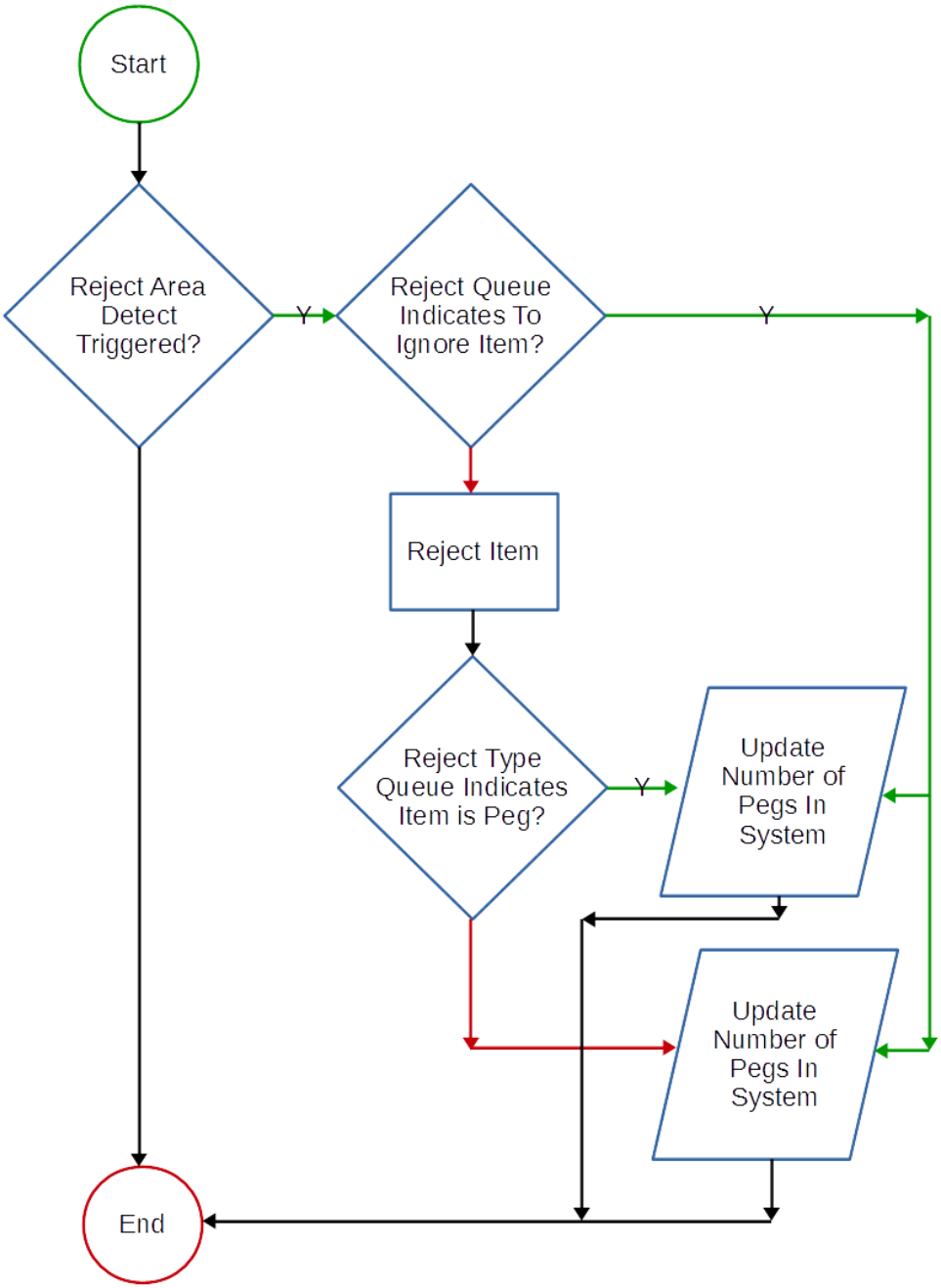

Figure 11 - Reject Area Handling

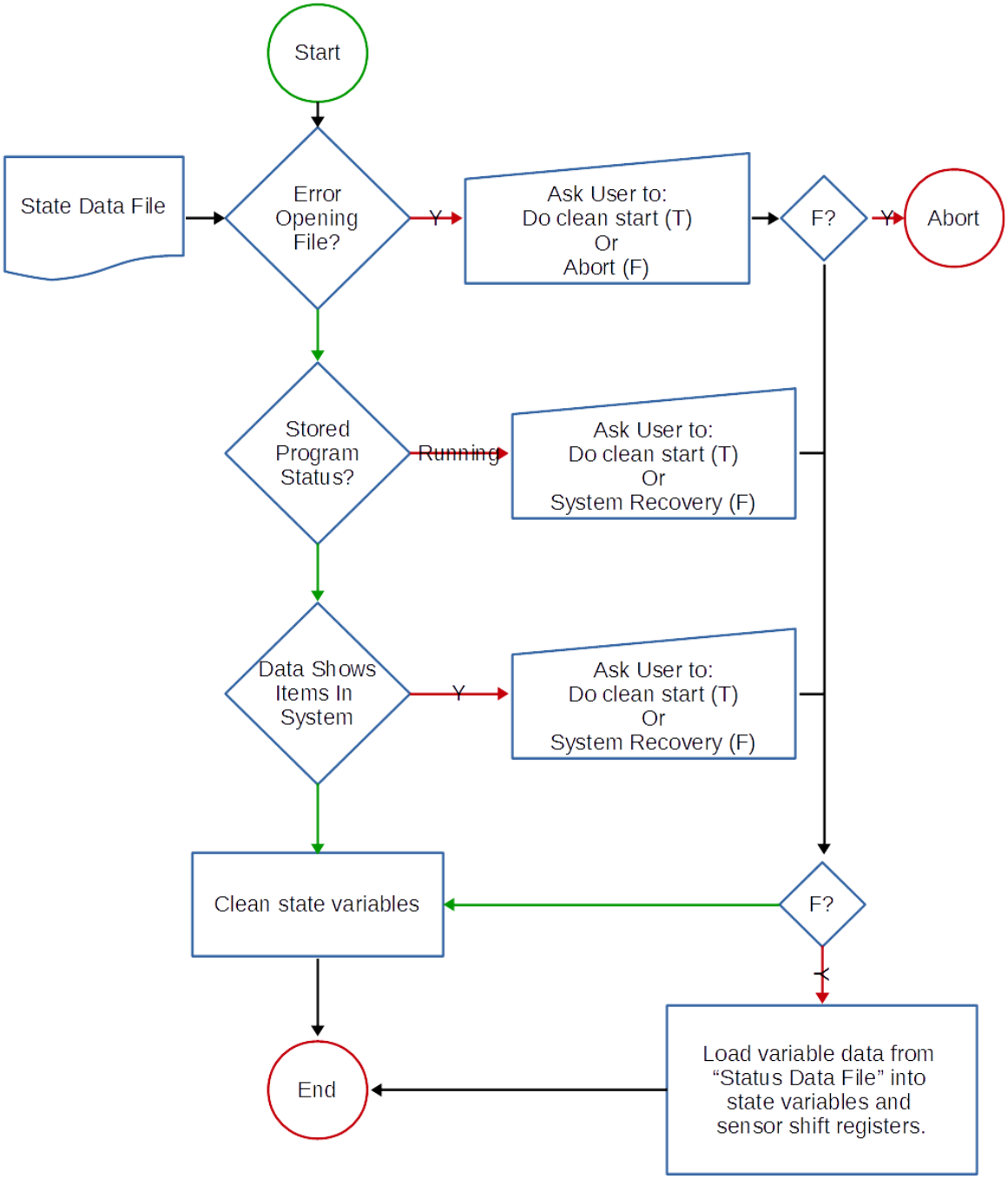

Figure 12 - Memory Recovery