COMPSYS 725 Distributed Cyber-Physical Systems Design
Department of Electrical, Computer, and Software Engineering

Assignment 2

Assessment percentage: 24% of the final grade
Assignment 2 (Part A) - Comparison Table & ECC Implementation: 6% of final grade
Assignment 2 (Part B) - Demo & Interview: 10% of final grade
Assignment 2 (Part B) - Final Report: 8% of final grade

I. Overview

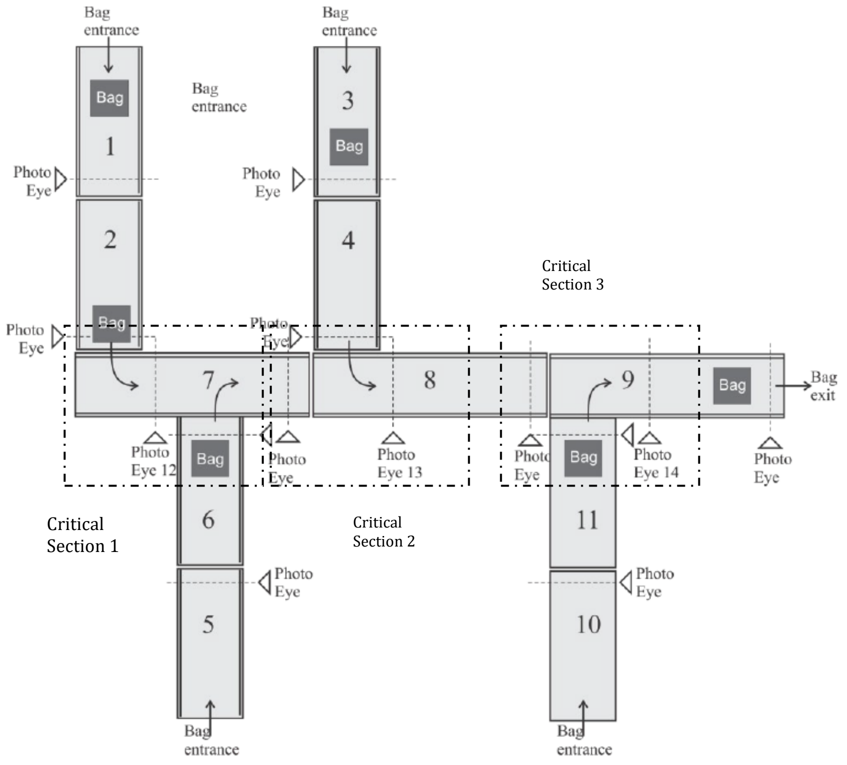

Figure 1: A small baggage handling system.

The company IdeaCorp has asked your team of 3 to improve the implementation of a rudimentary baggage handling system (BHS). In this assignment, you will document and implement the conveyor control system for a small part of the BHS. Your main aim is to prevent the collision of bags using appropriate mutual exclusion algorithms.

The system consists of 11 conveyor sections. Bags can appear at one of four entrances and should be moved to the bag exit. Conveyors are equipped with several Photo Eye (PE) sensors that can be used as inputs to the control logic. In this assignment, you will use a simulated environment running on a desktop PC; in the real system, each conveyor section can be controlled using an embedded microcontroller.

Baggage control is already implemented in a fully distributed fashion (with several re-used function blocks reading sensors and communicating with other controllers). Note that controllers are grouped as composite function blocks according to the grouping in the physical layout. The layout of the baggage handling system with the position of Photo Eyes, bag entrances and bag exits are shown in Figure 1.

I. System requirements

if 7 stops, 1,2 and 6,5 has to stop. If 5 sarts, 6 have to start.

1. The BHS layout has three critical sections, so the system must safely handle bag merging.
2. Conveyors should implement cascading start and stop. For example: when conveyor 7 stops, it should issue a stop to conveyors 2. 6, and conveyor 2 will issue a stop to conveyor 1, and similarly, conveyor 6 issues a stop to conveyor 5. When conveyor 4 stops, it should issue stop to conveyor 3 and vice versa for starting.

global: work out with team

II. Assignment Tasks

- Using the following mutual exclusion (ME) algorithms[1]: central server, ring token, and multicast, implement a mutual exclusion algorithm for each critical section. Use central server for Critical Section 1, ring token for Critical Section 2, and multicast for Critical Section 3. The three Critical Sections are marked in Figure 1[2].
- Also, implement cascading start/stop.
- Your group will need to implement mutual exclusion for the three critical sections and demonstrate this in the final interview.
- Each person in the group has to implement one critical section.
- The implementation *has to be based on functional blocks* and implementation of the ECC using actions (algorithms, events and guard conditions) compliant with the IEC 61499 standard. Do not implement the entire system using Java or other programming language-based approaches.
- The implementation should work during the demo, and you should be able to explain it. Marks will be deducted if the implementation does not work during the demo.

Submission 1: Assignment 2 (Part A) - Comparison Table & ECC Implementation:

- Section 1: Introduction to report
- Section 2: Present a table comparing the three mutual exclusion (Central Server, Ring Token, Ricart Agrawala) algorithms based on various metrics using lecture material AND your reading of the textbook chapter. Include all three ME properties and atleast two other criteria.
- Section 3: The assignment requires you to implement one ME algorithm. For the ME algorithm you have chosen to implement, prepare an execution control chart (ECC). The ECC need not be implemented on FBDK. It can be drawn on FBDK or any drawing tool.
- Section 4: Conclusion to the report
- Submit a PDF with the table and the ECC, with a basic explanation of the ECC. Page limit – 3 pages.

Submission 2: Assignment 2 (Part B) - Demo & Interview

- Your group will have an interview in Week 12, during which you will have the opportunity to present a demo of your system and explain the approach you have used.
- You will be asked questions about your specific implementation and your group's integration approach.
- The implementation should work during the demo. Marks will be deducted if the implementation does not work.
- Individual marks are allocated for you own ME algorithm implementation and questions that are asked to you. The group mark is allocated for your integration approach.
- The group's implementation has to be submitted as a zip folder. The changes can be made to the same fbdk.zip given to you, and the entire zip file with the entire system implementation should be submitted on Canvas.
- You must include a README file that explains what you have implemented, which algorithm for which section, and what steps are needed to get your assignment running.
- Only one submission per group is sufficient.

[1] These algorithms will be discussed in lectures, but you can also read about these directly in 15.2 of "Distributed Systems" (available in the Reading List on Canvas) if you want a head start.

[2] Use BaggageSystemCTL in the cs725 folder/package. You may also find it helpful to modify control logic of a single conveyor in ConveyorCTL function block to add the implementation of your designated mutual exclusion algorithm. The current code is basic and reads only one PE.

Submission 3: Assignment 2 (Part B) - Final Report

- Explain the implementation in your own words, including the ECC and how it works, referring to your screenshots as required. – explain all the new events, variables, states or function blocks you have introduced.
- Include a screenshot of the implemented ECC (and network of FBs if necessary) – it should have good resolution.
- Include an additional flow diagram (if your ECC is particularly hard to follow).
- End the report with a Discussion section (200 words). This section should include reflections on your implementation and how well your implementation met the mutual exclusion requirements.
- Add a picture of your favourite Engineering meme at the end of the report.
- Page limit – 5. This is to incorporate any figures you want to. You can use additional pages (more than 5) for figures alone if needed.
- In Part A report, you were given clear instructions on what each section of the report should. That was in preparation for Part B report. The Part B report includes additional events/variables/blocks, ECC explanation, Discussion about ME properties and reflection on the design. These essential components have to be organised into logical sections. 1% mark is for organisation of the report. Being able to present your design effectively, and in such a way that another person can reproduce it is a key aspect of report-writing.

The reports should be submitted as PDF documents.

Appendix B - Description of the Baggage Handling System

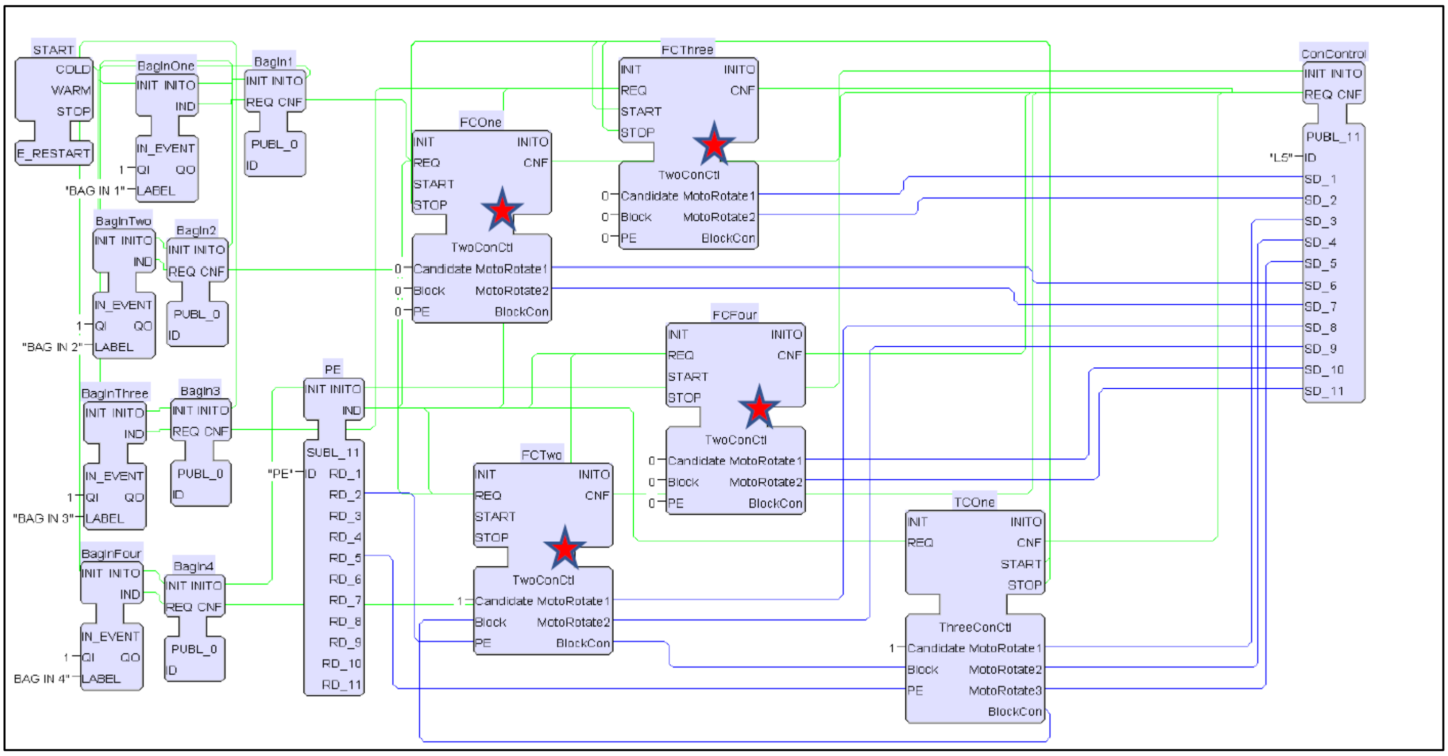

Figure 2: Function block diagram representation of the baggage handling system

The Baggage Handling System is included as a system file name BaggageSystemCTL in the cs725 folder in the fbdk.zip provided to you.

The system file has a VIEW device for visualization (of ImageDev type) and a HMI device (of FRAME_DEVICE type) for control and HMI.

The VIEW device has the following function blocks:

1. 4 instances of the TwoConveyor FB that Models the Conveyors 1 – 6 and 10, 11: Shows reusability.
2. 1 instance of the ThreeConveyor FB that Models the Conveyors 7 – 9
3. Each TwoConveyor FB controls the conveyor combination of two conveyors connected together and do not have any merge points.
4. The three conveyor FB controls three conveyor connected together and has four merge points.
5. The View block is responsible for the Visualization.
6. The ConControl SUBL blocks provide the input to the Conveyor models. Note that the only controllable input the conveyor is to turn on/off the motor. These inputs come from the distributed controller.
7. The PE PUBL block publishes the PE (Photo eye) values to the distributed controller.

HMI device has the following function blocks (as shown in Figure 2):

- 4 instances of the TwoConCtl FB that controls the Conveyors 1 – 6 and 10, 11 – Shows reusability.
- FCOne – Conveyor 1 and 2 (MotoRotate1 and MotoRotate2 respectively)
- FCTwo – Conveyor 3 and 4(MotoRotate1 and MotoRotate2 respectively)
- FCThree – Conveyor 5 and 6(MotoRotate1 and MotoRotate2 respectively)
- FCFour – Conveyor 10 and 11(MotoRotate1 and MotoRotate2 respectively)

Last updated: 6/09/2022 5:48 pm

- 1 instance of the ThreeConveyor FB that controls the Conveyors 7 – 9(MotoRotate1, MotoRotate2, MotoRotate3 respectively)
- The ConControl PUBL publish the motor rotate values to the conveyor models.

The PE SUBL block provides the PE input to the controllers. The PE FB outputs and the PE value they represent are

- RD_1 – PE 2
- RD_2 – PE 4
- RD_3 – PE 6
- RD_4 – PE 11
- RD_5 – PE 7
- RD_6 – PE 8
- RD_7 – PE 12
- RD_8 – PE 13
- RD_9 – PE 14
- RD_10 – Future Use
- RD_11 – Future Use

Note that the PE that don't have numbers in the figure means their number is same as the conveyor number that they are associated with.

The ConControl PUBL block is used to control the conveyor belts on/off and the Conveyor they represent are:

- SD_1 – Conveyor 5
- SD_2 – Conveyor 6
- SD_3 – Conveyor 7
- SD_4 – Conveyor 8
- SD_5 – Conveyor 9
- SD_6 – Conveyor 1
- SD_7 – Conveyor 2
- SD_8 – Conveyor 3
- SD_9 – Conveyor 4
- SD_10 – Conveyor 10
- SD_11 – Conveyor 11

The changes you make to the Baggage Handling System will be on the HMI device, as the HMI device includes the function blocks that control the system. The VIEW device is responsible only for the visualisation.

Appendix C – Additional information to get you started

- Most of the Baggage Handling System is already built for you, except the implementation of mutual exclusion in the critical sections. This implementation of mutual exclusion will require you to use the same basic function blocks and add more events and data variables to it, or add one or two additional function blocks to the existing system.

- As a first step, try to get the conveyors moving. Again, this will only require some connections to be made by you in addition to what is already there.

- The VIEW resource only controls what is seen in the output screen that appears – i.e. the conveyor belt diagram. So unless you want to change how the conveyor system "looks", there is no need to change the VIEW resource. All the changes dealing with controlling what happens in the critical sections must be done at the HMI device.

- You may find it useful to note down which Photo Eye (PE) connects to which conveyor, and this in turn connects to which function blocks.

- You can add multiple actions to a single state. You can also have actions without algorithms, or actions without events associated to them. Just right click on the state, and explore the options.

- The mutual exclusion implemented at a particular critical section can be generalised, i.e. it can be made general for any number of conveyors connected to the critical section. Or you can implement the mutual exclusion specifically for this BHS layout, i.e. you can specify that only two conveyors are entering Critical Section 1, hence we are designing the system to work for only two conveyor inputs. Please include your decision in your report if you decide to generalise or be specific.

- All the changes to the function blocks can be done using mouse clicks and selecting options. But the same changes can also be brought by editing the xml, as demonstrated in the lab. Remember to press ALT *before* trying to make connections between elements.

- If you made changes to the system configuration, and save the changes, you may have to re-start FBDK for the changes to be reflected in the output.

- If you make changes to any function blocks, or connections to function blocks in the system configuration, remember to *save and compile each of the function blocks* before saving the system configuration and restarting FBDK. Only if you *save and compile each function block will the changes be* saved to the system configuration.

- It may be a good idea to check the .fbt file to see if the changes you made have been saved before exiting FBDK.

- If you get unexpected errors like "No SUCH OBJECT" error for an object you have added (and you can see it in the .fbt file), then restarting FBDK can sometimes solve the issue.

- The first state you create in the ECC will appear with a double rectangle box – indicating that it is he first state for the function block.

- To make a new composite function block, first add/edit the basic function blocks that form the composite function block. This will involve editing the basic function block by adding or editing events or data variables. Once the basic function blocks are ready, edit the composite function block. Even though new events and data variables are added to the basic function blocks, they do not automatically appear in the composite function blocks. So, you have to add the events and data variables to the composite function block. Once these additions are done to the composite function block, all events in the basic function blocks have to be connected to the composite function blocks. This can be done by right-clicking each of the variables of the basic function blocks and selecting which variables of the composite function block you want to connect them to.

- Some events variables and their details:

  Event Inputs:
  INIT – Triggered at the startup of the application. Used to initialise the variables.
  REQ – Triggered every time there is a change in PE sensor value. Updates the information of the function block.
  CAS_START - Used to control when the conveyor starts.
  CAS_STOP - Used to control when the conveyor stops.

  Event Outputs:
  INITO – Cascading intialise signal to other conveyors.
  CNF – Indicates to motor controller when there is a new motor value.
  START – Indicates to other conveyors they should start.
  STOP - Indicates to other conveyors they should stop.