Cart LabVIEW Program Step-by-Step Tutorial

I. Purpose:

Based on the basic knowledge of LabVIEW introduced in the previous tutorial file and video, this tutorial details the process of writing the LabVIEW program of the whole cart. With a start of frame construction, adding loop, and basic blocks for testing the myRIO device connection, this tutorial lists all the steps and figures of writing the cart program.

This is a long tutorial that you may feel dizzy, but please be patient, you will enjoy the process and become a LabVIEW programming master after going through this tutorial. Have Fun!

II. Requirement:

1. Basic knowledge of LabVIEW programming (you can achieve that by reading the previous tutorial files and watch the YouTube video provided in the class)

2. The myRIO device

3. A fully built cart. You can have your TA help you to check if the cart is right for this program writing.

4. A computer with a full student version of LabVIEW from NI installed.

5. Connection cables and batteries

III. Process:

Chapter 1: Frame and Loop Construction

1. Open a new LabView project. You can see two windows popped up, one is called Front Panel (FP) and the other one is called Block Diagram (BD). In the previous tutorial, you should already know that the use of these two windows. The BD window is used for program writing, so this is the window you would mostly work on. If only the FP popped up, don’t panic, simply press “ctrl + E”, you can say hello to your BD window!

2. In the BD window, right click on anywhere, you should see a whole set of tool boxes popped up. Go to Express → Exec Control → Flat Sequence. Click on that, your mouse should become a cross mark. Left click on the upper left corner of the BD window and move your mouse all the way to the lower right corner and left click again. This is the frame that your program should be in. If you cannot find this, don’t worry, you can simply click on “search”, which locates on the upper right corner of the tool box list. Click on that and type in “Flat Sequence”, the Flat Sequence icon
should be coming right up. **Remember, you can always do the “search” if you cannot find anything in the tool box list.**

3. Right click on the frame of the Flat Sequence, and choose “Add Frame After” and “Add Frame Before”. Then the frame should be divided into three parts as shown in Fig. 1.

![Fig.1 Divided Flat Sequence Frame](image)

4. Under the same category (Express → Exec Control) you can also find the “While Loop”, please go ahead click on that and put it right in the middle of the “Flat Sequence” frame. You will put most of your blocks inside of the while loop frame. After you place the while loop in the BD window, you can go back to the FP window to find the “stop button”, which locates at the upper left corner of the FP window. The “stop” button is the main switch of the program, which allows you to stop at any time when the program is running.

5. Under “programming” category, you can find the “Wait (ms)” block in the “Timing” option. Put that at any corner within your “While loop” and find the “Numeric Constant” block and put it on the right of your “Wait (ms)” block. When you put your mouse on the right edge of the “Numeric Constant” block, you can see your mouse becomes a spool. That means the LabVIEW wants you to connect this block with any other ones. Go ahead connect the “Numeric Constant” with the “Wait (ms)” block. This step specifies the calculation sampling period of your discrete program. So please put a number in the “Numeric Constant” simply by double clicking on that block and type in a number (Let’s say 25). Then, your program will finish calculating the while loop every 25 ms. Congratulations! You have completed your first step of your LabVIEW programming.
Chapter 2: Connection Test Construction

In order to see if your program can communicate with the actuator (myRIO), you need to build a simple program to test the connection between your computer and myRIO.

6. Find the “Button” and “LED” blocks under “myRIO” category. Place those blocks inside of the “While Loop” with the “LED” block under the “Button” block.

7. Now, it’s time to design your User Interface layout. Right click on your FP and find the “Boolean” option. Under that, you can find “Push Button” and “Round LED”. You will need four of the “Push Button” and one “Round LED”. You can edit the name by double click on the name of the corresponding indicators (Round LED) or controls (Push Buttons). Put those things anywhere you want in the FP.

8. When you go back to your BD, you can find the corresponding block randomly placed in your “While Loop”. Connect the “Round LED” to the “value” outlet of the “Button” block, and connect all of your “Push Buttons” to the “LED #” of the “LED” block. After you doing this, your program should be able to communicate with the myRIO. But you also need to put the self-detect system into your program in case something goes wrong.

9. Right click on the BD again and find the “Error Cluster Constant” under “Programming → Dialog & User Interface”, and place the colorful block in the first frame of your “Flat Sequence”. Connect the “Error Cluster Constant” to your “Button” and “LED” with the “error in (no error)” inlet. Find the “Merge Errors” block under “Programming → Dialog & User Interface”. Drag a line from the “error out” of your “Button” and “LED” blocks to the left connector of the “Merge Errors”. Also, if anything is wrong, you may want to know what that is. You can put an error indicator that gives you error codes in the user interface. You can find that in “Classic → Array, Matrix & Cluster → Error Out.ctl” in the FP. Go back to your BD, put the corresponding “error out” block in the third section of your “Flat Sequence” frame and connect the “Merge Errors” to the “error out” block. Also, you need to wire the while loop condition to the error route. The condition block locates at the right lower corner of the “While loop”. Everything so far should be look like this:
You can connect any control function button to the “LED” block later so that the LEDs on the myRIO can be indicators of certain movement or condition.

Chapter 3: Cart Driving Programming

Before proceeding to this chapter, you can ask your TA to have your program checked. Making sure everything is perfectly right is essential before you take to the next step. Now, let’s run the cart!

The motors are driven by PWM signal. 100% PWM represents forward full speed, which is 200 rpm and 0% represents backward full speed at 200 rpm. When the PWM signal is 50%, the motor is supposed to be stopped. The PWM signal is output by the little board on the cart, which is called “sabretooth” or motor driver. MyRIO provide the motor driver with analog output and the motor driver can convert it into a corresponding PWM signal to drive the motor. So the program of driving the cart is written by using the “Analog Output” block in the LabVIEW.

10. You can find the “Analog Output” block in the category of “myRIO → Onboard”. You need to open the “Analog Output” block to define the output pin, which is the pin on myRIO device. Since we have two motors to drive, we need two “Analog Output” blocks and the pin setting should be different. Put two “Numeric Constants” in the “While Loop” and connect those to the “Analog Outputs” on the “A/AO#” connector. Check the motor wiring to see if you are connecting to the right pins.

11. Turn on the cart and the myRIO, give those “Numeric Constant” a number (70 is recommended). If motors are running at a constant speed that means you got it right.

12. Now we want to change the speed by using the user interface when the cart is running. Delete the “Numeric Constant” blocks that are connected to the “Analog Output” blocks. Go to the FP and find the “Tank” under “Modern” → “Numeric” option. This is the speed control unit we are going to use to set the linear speed of the cart. You can also use something else under the same category, as long as it has
changeable numeric value. Since the “Tank” block is set to be an indicator by default. Right click on that and choose “Change to control”. Connect the “Tank” block to the “Analog Output” blocks. Now the cart should be able to run at any speed you set in the user interface.

Chapter 4: Cart Encoder Programming

Before we make the program more complicated, we need to know how fast the cart is running, that means we need set up the encoder system. In the lab pre-lab, the principle of encoder is given. Make sure you read through that before you write this program.

13. In the “myRIO” category, you can find the block named “Encoder”. Put two of that in the “While Loop” and set the input pins by double click on the “Encoder” blocks. The pin setting of the “Encoder” blocks should also be different.

14. In the FP, find the “Numeric Indicator” in the “Modern” → “Numeric” option. You can change their name to “Counter Value L (or R)” and connect it to the corresponding “Encoder” blocks. After doing this, when the cart is running, the counter value on the user interface should be growing very fast. The reason is explained in the pre-lab. We are going to use this value to calculate the current motor speed.

15. Once the counter value is set up, we are able to set up the wheel current speed. Since the whole system is running in discrete time, and the sampling period is set to 0.025s (see step 5). The current speed of a wheel can be considered as the growth speed of the counter value, which can be achieved by subtracting the counter value in the previous sampling period. Find the “DBL Numeric Constant” under “Programming” → “Numeric” and put that in the first section of the “Flat Sequence” and set it to “0”. Find the “Subtract” under the same category. Connect the upper connector of the “Subtract” to the counter value output and lower connector to the “DBL Numeric Constant”.

16. After the connection is done, you need to tell the computer that the “DBL Numeric Constant” is the previous value of the counter value. That is very simple. Along the connection line, you can right click on the node locates on the “While Loop” and chose “Replace with Shift Register”. After you do that, there is an exactly the same node appears on the other side of the “While Loop”. Connect your counter value output to the node on the other side.

17. You also need to reset the counter value every cycle to achieve the current speed. Find the “True Constant” locates in the “Boolean” category and put it in the first section of the “Flat Sequence”. Connect that with your “Reset Counter” input of the
“Encoder” block. And change the node type to Shift Register by doing the same as step 16. Put a “False Constant” in the “While loop” and connect that to the Shift Register on the other side.

18. In the FP window, add another “Tank” to indicate the wheel speed and in the BD window connect that to the system. You also need to convert your counter value to speed (rpm), the calculation process is shown in the following figure. So far, the cart should be able to run at any speed and you can also observe the speed value.

![Fig. 3 Counter value convert to rpm](image)

Chapter 5: Cart Angular Speed Programming

Now, we want to control the linear speed and angular speed of the vehicle at the same time. The linear and angular speed of the cart are controlled by the speed of the motor. By giving different speeds of the two motors, the turning can be completed. The equations of the angular and linear speeds are given in the pre-lab. All you need to do is to set up the layout of your user interface and write the program according to the equations.

19. In the FP window, drag a “Knob” under “Numeric” category to control the direction of the cart. The corresponding block should be appeared in the BD window. You can build the logic relations according to the equations provided in the pre-lab.

Chapter 6: Cart PI Control Programming

Here comes the final step: PI Control. With the PI control, the cart is able to maintain a constant speed when it is making turns. The PI control program is written in discrete time domain. The equations are introduced in the pre-lab. You can finish the controller design by your-self or follow the figure provided below to finish the program.
20. In order to observe the motor response due to the PI controller, you may need to put a graph in the program. You can find the layout of putting the graph in Fig.4.