XY-Stage Application Manual

1. Starting window:

Upon starting the main file, the user interface will be open:

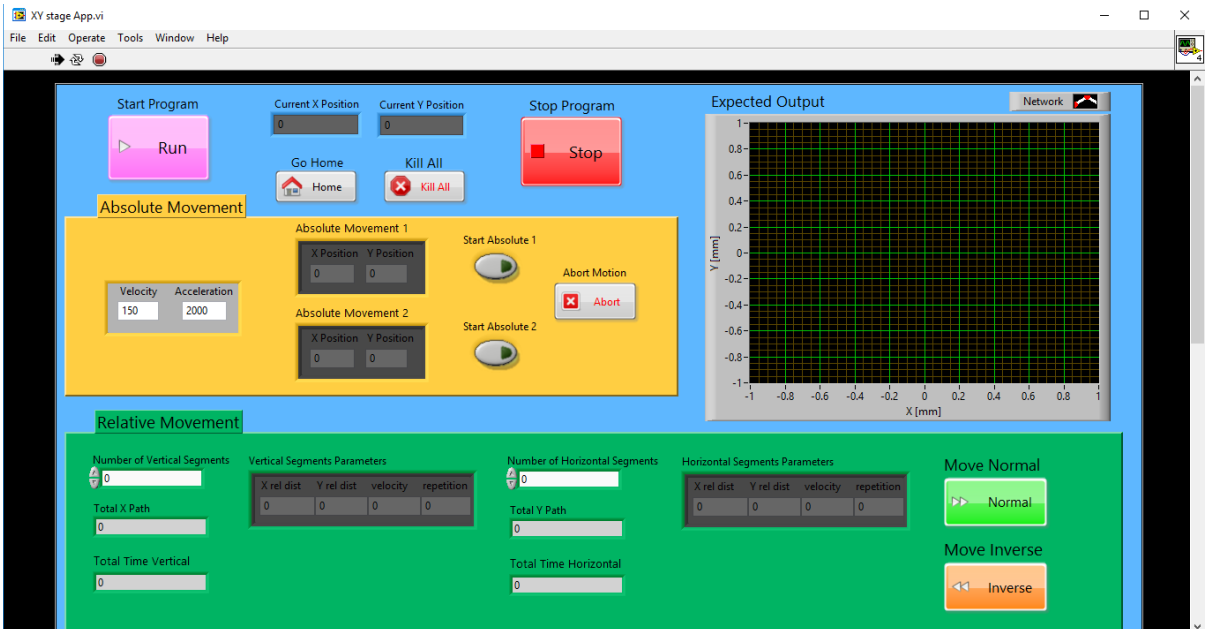

Figure 1: Startup window

- Note that on startup the arrow symbol on the top left is black (**Fig 2 a.**), meaning the application is running. If at any point the arrow turns white (**Fig 2 b.**) click on it to enable the application again (e.g., after pressing the "Stop Program" button).

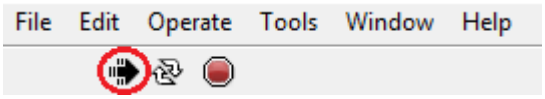

Figure 2 a.: Application is running

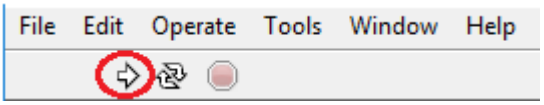

Figure 2 b.: Application is not running

At that point, the application is running, but the motors are not yet initialized and ready to work.

2. Front panel:

The front panel is generally split into 4 sections:

I. Command and Display:

This area is used for general controllability and monitoring of the motored stage.

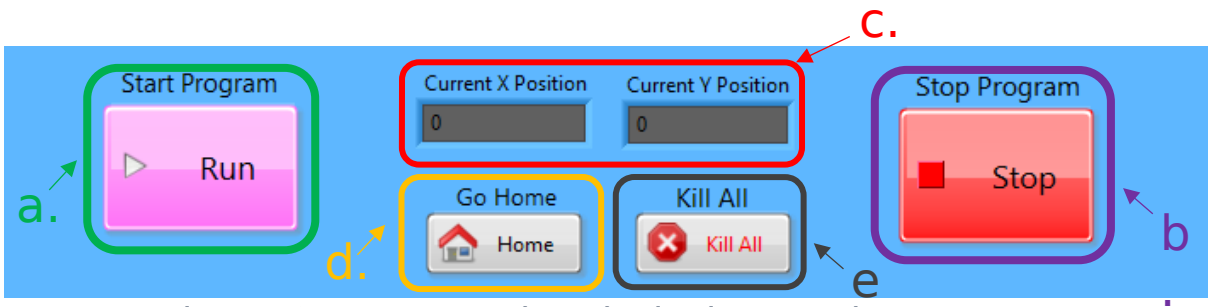

Figure 3: Command and Display section

a. Initiate the motors and sends the stage to position (0,0). To start commanding the program the motors has to be initiated.
b. Stops the program from running (changes the arrow in toolbar). It is recommended to stop the program before quitting the application.
c. Display the absolute position of the stage.
d. Sends the stage home which is defined as the absolute position (0,0).
e. Shuts the motors off but **do not** stops the program from running. It is necessary to shut the motors off before stopping the program.

II. Absolute Movement:

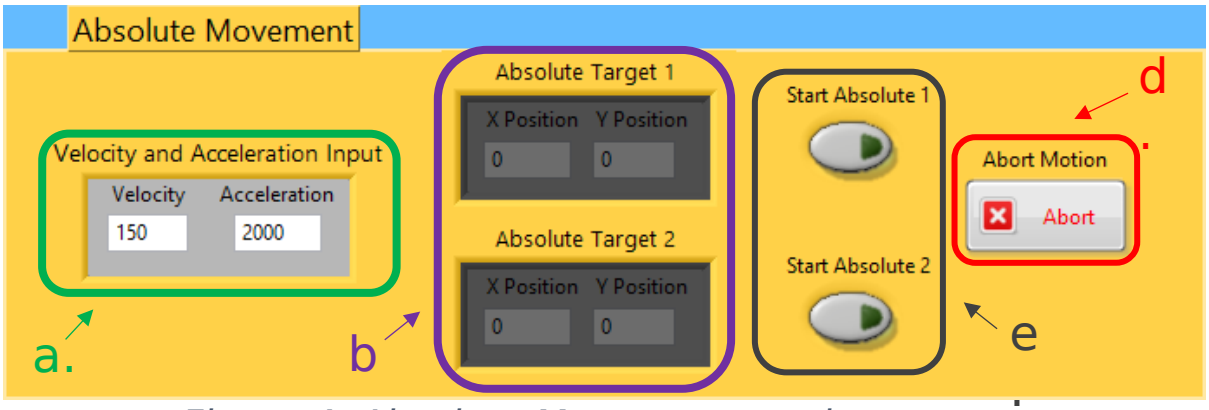

Figure 4: Absolute Movement section

a. Velocity and Acceleration input for the absolute section. The inserted values are set and being used for absolute movement of both axes.

b. Input of the desired target absolute position. There are 2 independent sets of target positions for convenience.

c. Execute the absolute movement of the corresponding set from b., with the velocity and acceleration from a.

d. Abort a job in action. This button aborts any job while running (including from other sections). It is placed here purely for esthetical reasons.

III. Relative Movement:

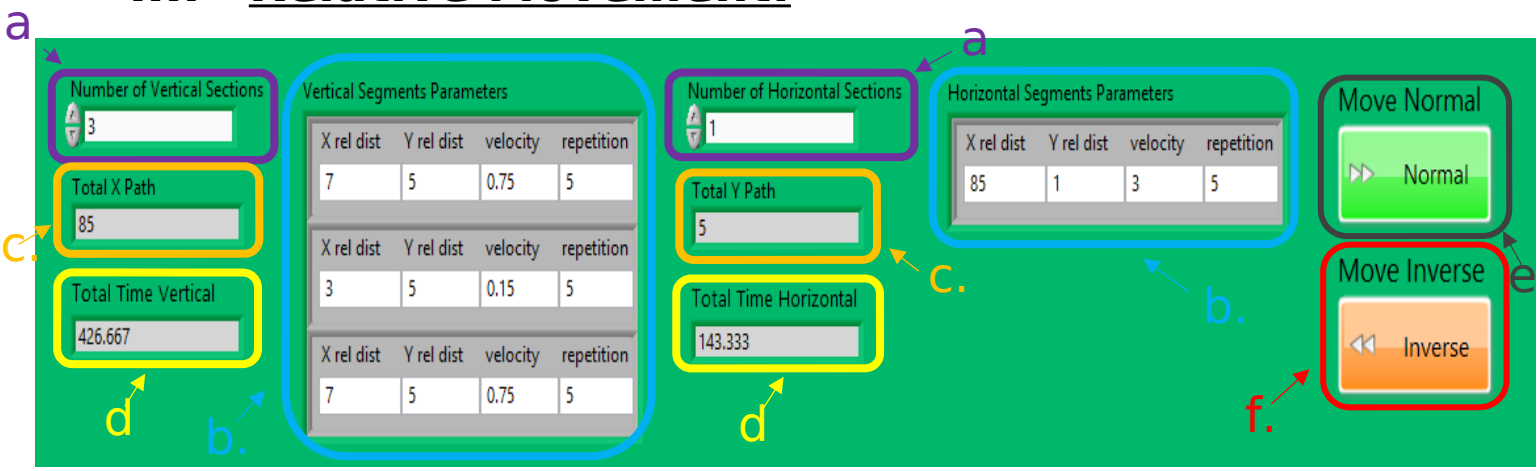

Figure 5: 3D Print Mode

The 3D print mode was designed to produce networks according to user specifications. It is set

to move in “L” shape segments with dimensions specified by the user to produce a network architecture. The program assumes the production starting point is the lower left point of the net. The parameters available to edit are divided to vertical and horizontal segments to extend controllability over the networks’ spatial density. The vertical segments are executed first one registry after another, and then the horizontal segments executed in the same manner.

Example:

The output of the parameters specified in **Fig 5** is shown in **Fig 6b.** in a graph. The starting point of the production is the lower left point, specified as (0,0) (it is **not** the absolute position of the stage at the starting point). The first row is executed 5 times producing “L” shape segments that are 5mm long and 7mm wide. In the 1st repetition the stage moves 5mm up, then 7mm to the right, 2nd repetition the stage moves 5mm down, then 7 to the right and so on… The horizontal segments function in the same manner.

a. Number of vertical/horizontal sections – specifies how many uniform sections are desired. This is not specifying the number of “L” segments, but how many different density sections are needed.

b. Vertical/horizontal segments’ parameters – directly specifies the parameters for the “L” shape movement within a section.

X, Y rel dist - determine the x, y size of the “L”.

Velocity – determine the stage’s velocity in this section for the entire movement.

Repetition – specifies the number of “L”s in this section.

c. Total X/Y path – shows the total X/Y path the user specified. It is helpful to know the total X movement done in the vertical path to specify the same length in the X parameter of the horizontal section.

d. Total time vertical/horizontal – Shows the total execution time expected for the entire vertical/horizontal movement (including all sections).

e. Move Normal – Execute the user’s parameters exactly as inserted.

f. Move Inverse – Execute the user’s parameter with a slight change: The X distance on the vertical segment’s parameter is multiplying by (-1), making it move in the other direction then specified. This is useful for linear spinning.

IV. Output Graph:

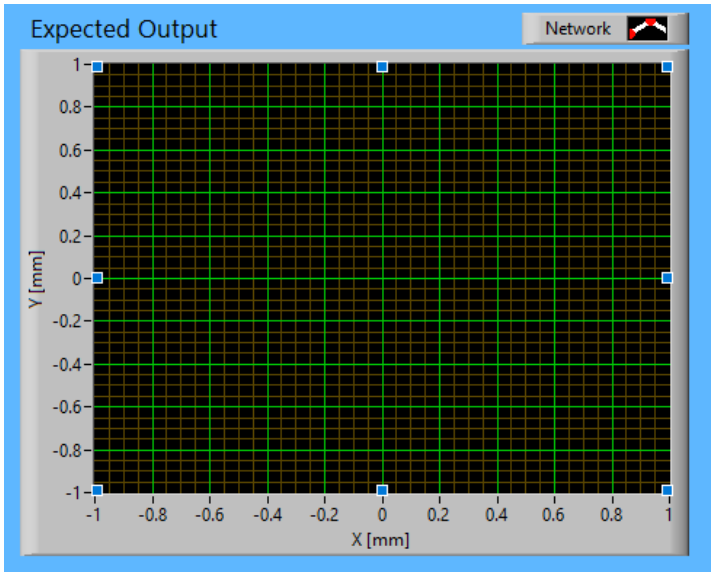

Figure 6.a: Empty Output graph

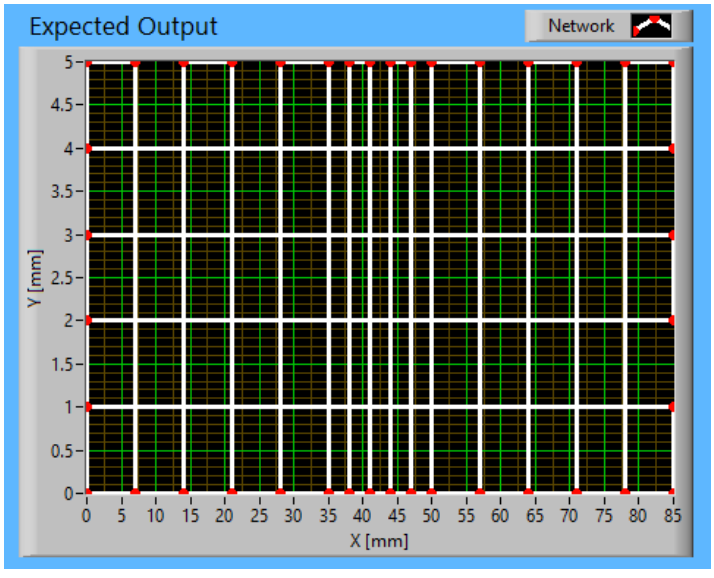

Figure 6.b: Expected Output based of input from input relative - 3D print

The graph shows the motion that stage is destined to make according to the input from the Relative Movement 3D print section.
Note that the graph only displays the number of segments stated by the user. So, for example when 0 horizontal segments are chosen, the first row of the input segment is displayed, however the expected output graph will not show any horizontal segments.
The white lines are representing fibers, and the red dots represent a change in movement direction by the stage.

4. Further Development of the Application:

The program is written in LabView and exported to an .exe file using the program (Tools --> Build Application From VI).

- General Architecture:
  The Block diagram (back-end) is built of one greater while loop with a case structure that contain everything. That allows the program keep running (white arrow in toolbar) while still in ideal mode (no connection to the stage). When the "Start" button is pressed the real bit of code starts running, while the "Stop" button shoots the user out of the greater while loop and ends the program's current run.
  Inside this greater loop + case structure, 2 parallel while loops are running – one (lower loop) contains most active user commands as "Go Home", "Move Absolute", "Move Relative", etc., while the other

(upper loop) checks constantly for stage position and allows abortion of in process jobs at any point.

In general, a minimum amount of code is intended to be done on the main VI, calling SubVis to do the job behind the scenes.
The usage of SubVis and their division into small chunks of code that serve for one purpose only in the most generic way possible is extremely recommended. SubVis are basically functions and should be thought of as such. Only sequences & structure loops, front panel objects + their properties, and minimum number of checks that relate to front panel objects should be done on the main VI. It is recommended to try keeping it clean.