LAB FINAL PROJECT

Team Members:
Antonieta Della Sala Pereira
Laiba Aurangzeb
Cagla Unsalan
Maria Alejandra Villarreal Valenzuela

With the program created in Lab 8, create a sequence including at least 10 Robotargets (points).

Report:

This project introduced us to the fundamentals of robot programming and motion control using ABB's RobotStudio environment in combination with the FlexPendant interface. The main objective was to define 10 different target points (robotargets) and move the IRB 1090 robot within those points by developing a basic movement program using the ABB's proprietary programming language, RAPID. This hands-on activity provided practical experience with motion commands and gave us insight into how robotic instructions are sequenced and executed.

We started off with downloading the ABB RobotStudio software on our computer and installing the Omnicore controller and latest Robot Ware as the add-ins to this software. Then we created a station and added the IRB 1090 robot from the ARB library to our station. Then we selected 'From Layout' option from the 'Virtual Controller' dropdown so that the software identifies the specific controller that is compatible with the robot we added to the station. Then we selected the 'Omnicore Flexpendant' under the 'Flexpendant' option of the 'Controller' tab. This opened a screen which is the same as the Flexpendant screen. After turning on the motor under the 'Jog' option, we got a virtual controller that is equivalent of the controller switch on the physical Flexpendant and could be used to manually move the robot in the following 3 planes: x, y, and z.

Before creating a program, we first ensured that the robot was in its home position i.e., it has coordinates (0,0,0). We then proceeded with creating a new program by selecting the 'Code' option in the FlexPendant interface, choosing the 'New Program' and then selecting the 'Main Module' option to allow us to start coding. The 'Add Instructions' option in the main module allowed us to select the different movements of the robot with the most used instructions being 'MoveJ', 'MoveL', and 'MoveC'. Using these movements,

we created 10 different instructions. For each movement instruction, we had to define the following 4 variables:

1. Target Point: The point in 3D space we want the robot to move to. It is a position variable which includes both location and orientation (robotarget).
2. Speed: The speed with which the robot should move towards the target point.
3. Zone: The precision of the movement or blending zone (e.g., 'zone = fine' means that the robot moves exactly to the target point)
4. Tool: This is the tool at the robot head used during robot's movement.

Before creating each new instruction, we had to use the 'Jog' control to move the robot to a new target point (robotarget). The coordinates of the new point (robotarget) were automatically copied to the coding module and so we had to then input the remaining 3 variables: speed, zone, tool. Once all 10 instructions were created to move the robot between 10 different target points (robotargets), we selected the 'PP to Main' option under the 'Move Program Pointer' dropdown of the 'Debug' tab of our coding screen. This allowed the program pointer to be reset to the beginning of the program so that the program is executed from the first instruction. We then proceeded to play this program and observed the continuous robot movement between different points as directed by the program pointer moving between instructions.

Our code:

```
MODULE MainModule
PROC main ( )
        MoverJ*, v500, fine, tool0;
        MoverJ*, v500, z10, tool0;
        MoverJ*, v500, z5, tool0;
        MoveL*, v500, fine, tool0;
        MoveJ*, v600, z30, tool0;
        MoveJ*, v600, z20, tool0;
        MoveJ*, v600, z10, tool0;
        MoveL*, v600, z5, tool0;
        MoveL*, v800, z5, tool0;
        MoveL*, v800, fine, tool0;
ENDPROC
ENDMODULE
```

Explanation of our code:

Our code begins with 'PROC main()' which means the sequences of events we have defined are structured as the main procedure. Each of the instructions begins with a variable that defines the robot's movement. The first few instructions begin with the

variable ‘MoveJ’ which represents the joint movement of the robot i.e., the robot moves all its joints simultaneously to reach the target point (robotarget). The ‘MoveL’ variable in the fourth instruction represents the linear movement of the robot i.e., the robot moves in a straight line to the target. The * indicates the target point (robotarget) for each instruction which we defined ourselves by using the ‘Jog’ to move the robot to the target point. The second variable in each of our instructions e.g., ‘v500’, ‘v600’ and ‘v800’ represents the speed in (mm/s) with which the robot moves towards each target point. The third variable represent the zone/ blending or stopping accuracy of the robot. In our instructions, we used five different types of zones i.e. ‘fine’ which causes the robot to stop exactly at the target point without any blending whereas ‘z5’, ‘z10’, ‘z20’, and ‘z30’ represent the blending radius through which the robot will blend the path instead of stopping precisely at the target. The ‘tool0’ in each instruction is the default tool that the robot is using. The sequence of events ends with the ‘ENDPROC’ code which means end of the procedure followed by the ending of the module.

The following is a summary of our code:

Our code is a combination of joint and linear movements while using both precise and blended targets.

1. MoveJ (joint movement) to first target point — precise stop as defined by ‘fine’.
2. MoveJ to second and third points — blended moves (smooth motion) as defined by ‘z10’ and ‘z5’.
3. MoveL (linear movement) to fourth point — precise stop.
4. MoveJ to fifth, sixth and seventh target points — blended joint movement again (with an increased speed of 600mm/s from the previous robotic speed of 500mm/s).
5. MoveL to eight and ninth points — linear blended moves.
6. MoveL to last target point – linear precise stop (with an increased speed of 800mm/s from the previous robotic speed of 600mm/s).

Link to our YouTube video showing how our program runs in ROBOT STUDIO:

https://youtu.be/tssr5Axr8LY?si=KUf3ufZqcbz-nNWc