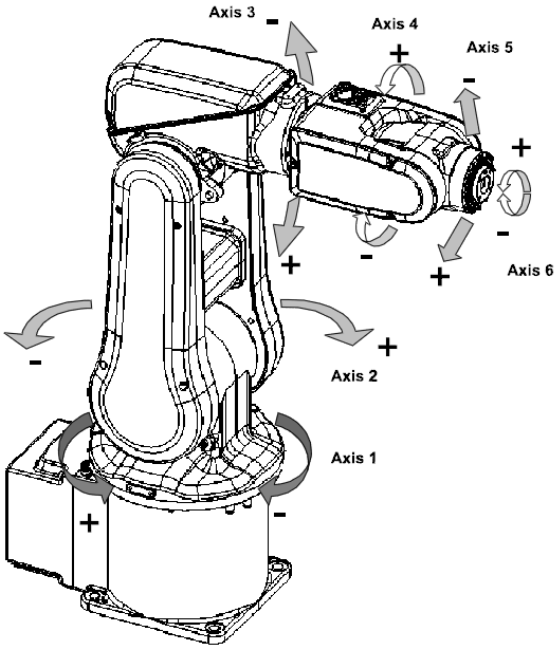
Drexel University
College of Engineering
Division of Management and Technology
Department of Engineering Technology
MET 205 Robotics and Mechatronics
Lab 4 Robot ABB IRB 120

Objective:

1. To learn the basics of a robot (ABB IRB 120)
2. To learn about the robot controller.
3. To learn to teach points.
4. To write a simple program.

Introduction

The IRB 120 is one of ABB Robotics generation of 6-axis industrial robots designed specifically for manufacturing industries that use flexible robot based automation. The robot has an open structure that is especially adapted for flexible use, and can communicate extensively with external systems.

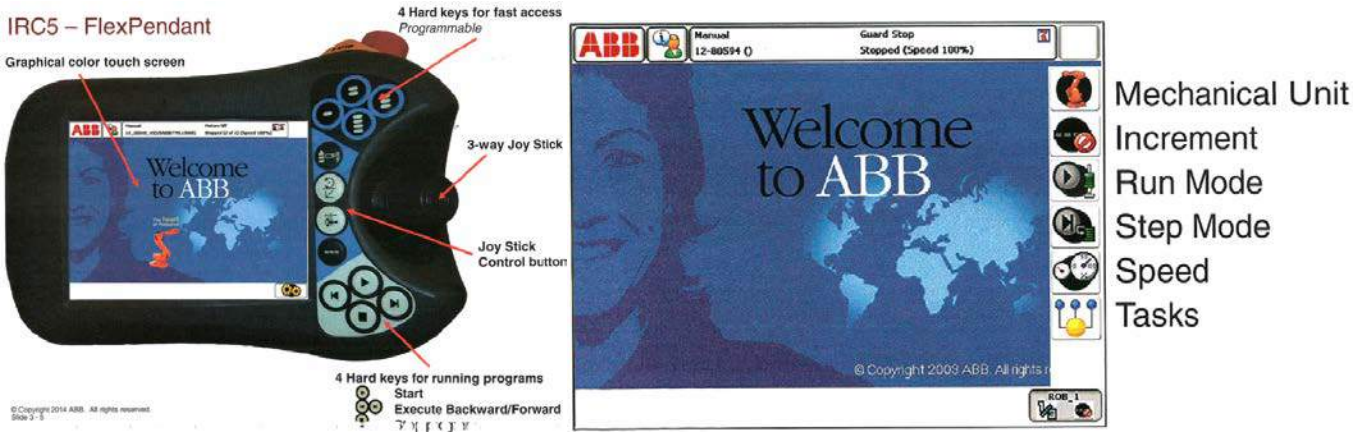


Operating system

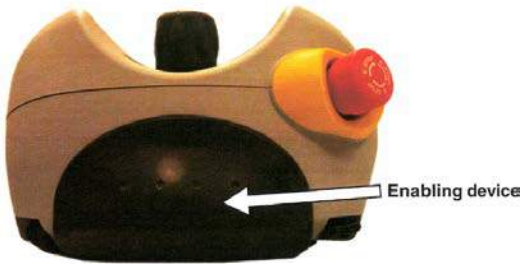
The robot is equipped with the IRC5 controller and robot control software, RobotWare for M2004. RobotWare supports every aspect of the robot system, such as motion control, development and execution of application programs, communication etc. See Product specification - Controller IRC5 with FlexPendant. Safety standards require a controller to be connected to the robot. For additional functionality, the robot can be equipped with optional software for application support - for example communication features - network communication - and advanced functions such as multitasking etc.

System Description of ABB IRB 120/Articulated:

Understanding the FlexPendant application life cycle improves your ability to design and debug the application. IRC5 is ABB's new generation robot controller. Virtual robot technology makes it possible to run a virtual IRC5 controller, virtual mechanical units and a virtual FlexPendant on the desktop. FlexPendant is ABB's new generation hand-held device, used with the IRC5 robot controller. It is developed with Microsoft's latest technology for embedded systems, Windows CE and .NET Compact Framework. The FlexPendant is a "smart device" in the .NET vocabulary, i.e. a complete computer in itself with its own processor, operating system etc.



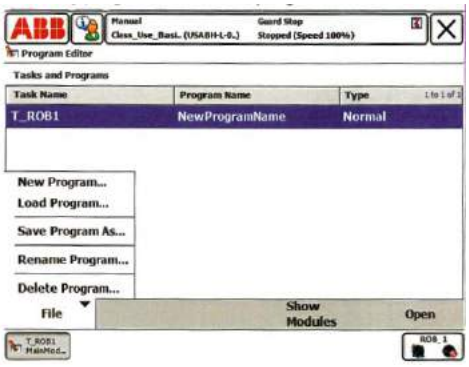
The enabling device is a pressure switch with three positions
The switch must be in the middle position in order to activate the motors
All root movement will immediately stop if the switch is released or pressed to the bottom



Saving and Loading a program

To create a new program or load an existing program or save a program:

Tap ABB
Tap Program Editor
Tap Tasks and Programs
Tap File
Tap New Program:
To create a new program
Tap Load Program:
To load an existing program
Tap Save Program as:
To save a program



Saving a program

A folder with the program name is created

Module: MainModule

File extension: pgf is an XML file that points to the Main Module and all other program modules (.mod).

IRC5 Program File structure

Folder NewTask Name

Active RAM

MainModule.**mod**

```
MODULE MainModule
  Data Declations: Tool Data,
  Robtarget data
  PROC main()
    Rotuine1;
    Routine2;
  ENDPROC

  PROC Routine1()
    MoveL p1, v1000, z10, Tool0;
  ENDPROC

  PROC Routine2()
    MoveL p2, v1000, z10, Tool0;
  ENDPROC
ENDMODULE
```

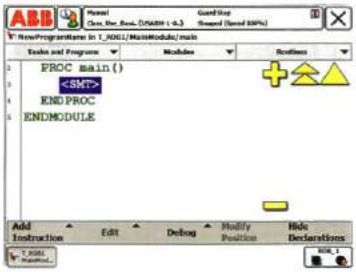
ModuleA.**mod**

```
MODULE ModuleA
  Data Declations: Tool Data,
  Robtarget data
  PROC RoutineA1()
    MoveL p3, v1000, z10, Tool0;
    MoveL p4, v1000, z10, Tool0;
  ENDPROC
ENDMODULE
```

Create a program

To Create a new program: (If no program exists)

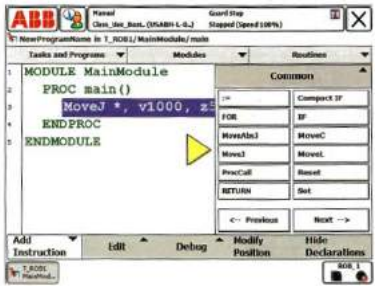
- Tap ABB
- Tap Program Editor
- Select Robot Task
- Tap New.



Inserting Move Instructions

To add instructions to your program:

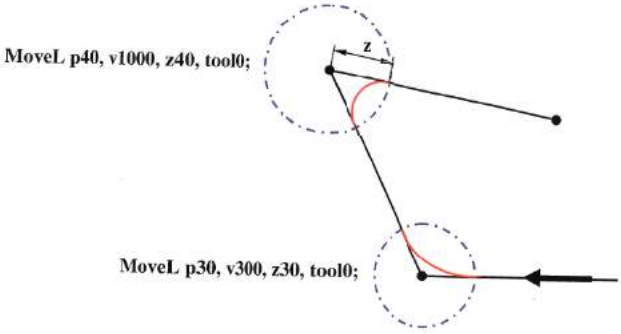
- Tap Add Instruction
- Jog robot into position
- Tap MoveJ or MoveL



Inserting Move Instructions

To name the position.
Double tap the ★ and press new.
Change the name by pressing the ... Box then use the key board to give the position a unique name for the position jog the robot to the next position and repeat.

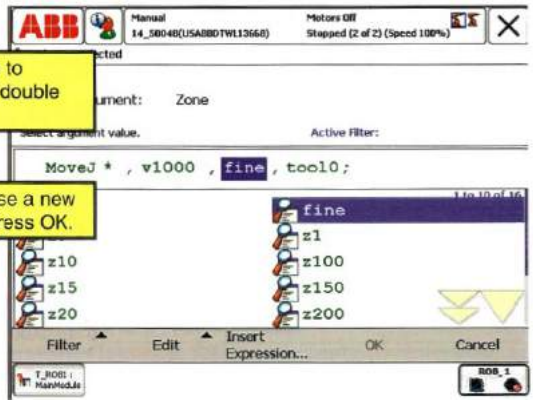
Velocity and Zones



Changing a data

Select item to change by double Tapping.

Then choose a new data and press OK.

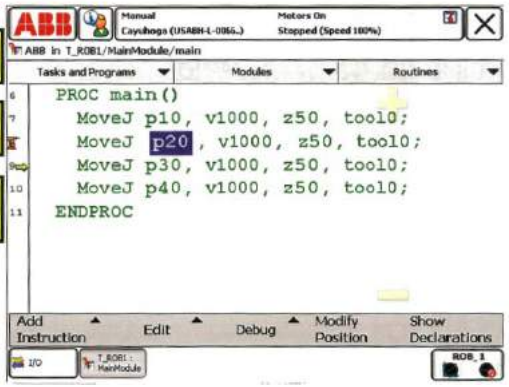


Modifying a Position

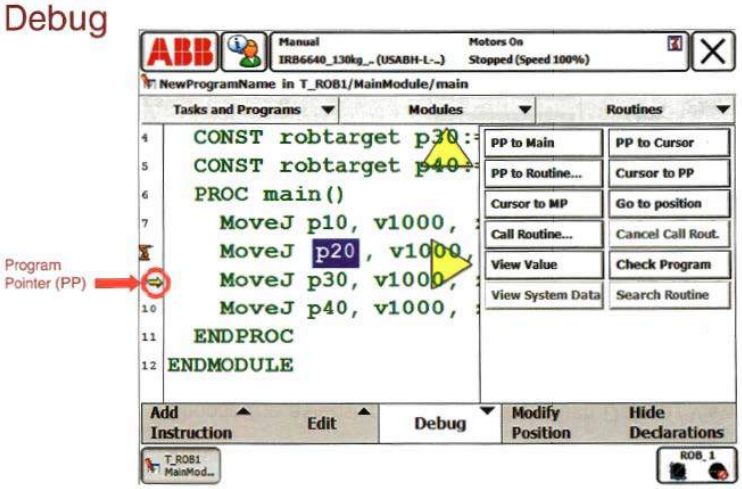
Step to the move to be modified.

Jog the robot to the new position.

Tap
Modify Position



Debug



Checking Robot Calibration

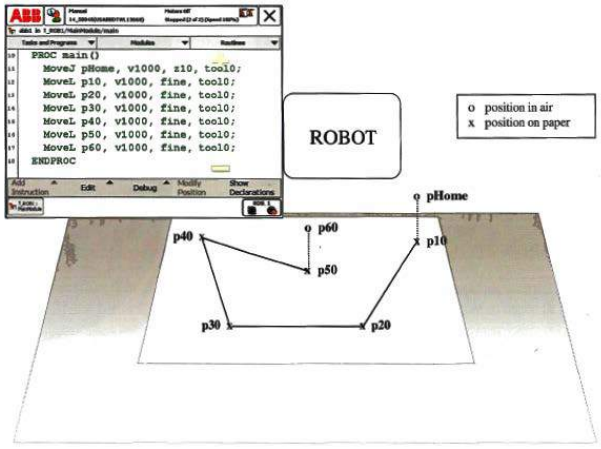
MoveAbsJ

Create a new routine (GotoCalib)

Insert MoveAbsJ instruction

Choose the asterisk position and then push Debug / View Value, put all 6 axis to zero.

Move Instructions



Stepping Instruction by Instruction

In Manual Mode, the routine may be executed step-by-step Line by Line forwards or backwards.

There are a number of dedicated motion buttons on the FlexPendant

Some Common Commands:-

1. MOVE Statement:
   *MoveC M*oves along a circular path
   MoveJ Joint movement
   *MoveL M*oves along a linear path
   MoveAbsJ Absolute joint movement
2. SPEED Statement
   Syntax: V<expression>: The velocity of the tool center point is expressed in mm/s (in the object coordinate system).

Requirements: 1. Create a simple program as described in the handout. Test your program for robotic calibration in XYZ coordinate.
2. Gently but firmly mount a pencil or marker on the robot gripper. Jog the robot to a position in which the pencil point touches a corner point of a square 50mm x 50mm. Move the robot to all the corner points (actual) using FlexPendant. Record the positions (X, Y, Z) the robot moves.
3. Try four different speeds: 125, 250, 500, and 1000 mm/s with Zone = 0 for running the program and record the cycle time for each test.
4. Plot a graph of cycle time vs. speed (mm/s).
5. Address the difference between the actual cycle time and the calculated cycle time.