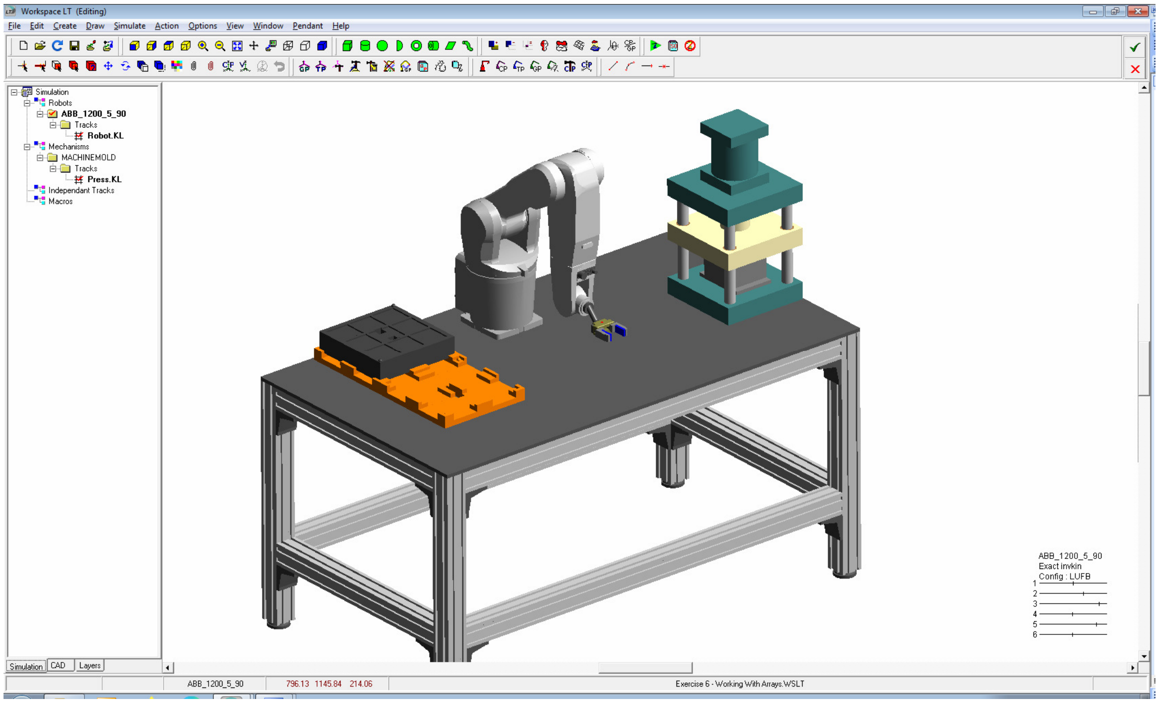
Workspace ROBOT SIMULATION LT



Exercise 6 - Working With Arrays

Objectives:

- Create basic robot programs in simulation.
- Understanding Logic statements.
- Using an end of arm gripper to manipulate parts.
- Using integers to create a three dimensional matrix.
- Using routines to assign various robot tasks.
- Integrate a robot, mold press, staging fixture and 3 pallets in simulation.
- Use of Inputs and outputs to control the simulation flow.
- Create an Animation file (.avi).

Materials;

- Workspace LT© simulation software.
- Workspace LT© project file Exercise 6 - Working With Arrays.WSLT
- Manual "Workspace LT© User Guide.pdf".

Helpful Hint; Before starting this exercise, review the User Guide sections;

- 6.3 - Comments and Workspace commands
- 6.4.5 Variables
- 6.7.1 - INTEGER and REAL functions
- 8.2.1 DIN..DOUT
- 9.1.1 - IF ... ENDIF
- 9.2.3 - WHILE ... ENDWHILE
- 12.1.2 Recording an animation
- 14.15 - Creating Mechanisms

1) Procedure: Create the Robot track
   a) Open Workspace LT simulation software.
   b) Open the project file "Exercise 6 - Working With Arrays.WSLT ".
   c) Add a track for robot ABB_1200_5_90. Enter the Track Name "Robot" and select the Language "KAREL 2". Using the mouse select OK when completed.
   d) Begin recording a track using the newly created "Robot.KL".
   e) The Action ☒ menu appears, select Begin to start recording.
   f) The Action ☒ menu alters the command options.
   g) Record the tool frame at POS(-45,0,134,0,0,0,'').

h) From the Action ⊠ Menu box, select Robot Move Commands. Once selected the Robot Move Commands ⊠ menu appears.

i) Using the mouse ↖, select the GP labeled "HOMEGP". Select the command GP Move. Although this Project's robot has a predefined home position use the GP labeled "HOMEGP" as home. Confirmation will be require as is true for all GP moves during a track recording. This move has now been recorded in the Track.

j) Using the mouse ↖, select the GP labeled "PRESSUNLOAD". Select the command GP Move. Confirmation will be require select OK. This move has now been recorded in the Track.

k) Select the command Move Relative. The "Relative" dialogue box will appear, enter the values X = 0, Y = 0, Z = 200. select OK to accept. The robot will move from "PRESSUNLOAD " positioning the gripper now ready for the object grasp. The actions have now been recorded.

l) From the Action ⊠ menu select Gripper Commands, the Gripper Com... ⊠ menu will appear. From this menu box select Close Hand and a Confirm ☒ menu box will appear, select No. The gripper will now close and no object will be attached to the robot face plate. The actions have now been recorded.

m) Select the command Move Relative. The "Relative" dialogue box will appear, enter the values X = 0, Y = 0, Z = 10 and select OK to accept. The robot will raise the gripper vertically. If there was an object "PART" attached, the simulation would show this object lifting to clear the mold. The actions have now been recorded.

n) Select the command Move Relative. The "Relative" dialogue box will appear, enter the values X = 0, Y = 0, Z = -200, select OK to accept. The robot will move clear of the press.

o) Using the mouse ↖, select the GP labeled "HOMEGP". Select the command GP Move. Select OK to accept. The robot will move to "HOMEGP". The actions have now been recorded.

p) Using the mouse ↖, select the GP labeled "PRESSUNLOAD". Select the command GP Move. Confirmation will be require select OK. This move has now been recorded in the Track.

q) Using the mouse ↖, select the object labeled "PALLETCLRGP". Select the command GP Move. Select OK to accept. The robot will move to " PALLETCLRGP ". The actions have now been recorded.

r) Select the command Move Relative. The "Relative" dialogue box will appear, enter the values X = 200, Y = -200, Z = 0, select OK to accept. The robot will move above the pallet, part nest. The actions have now been recorded.

s) Select the command Move Relative. The "Relative" dialogue box will appear, enter the values X = 0, Y = 0, Z = -100, select OK to accept. The robot will move to the release position of the object PART. The actions have now been recorded.

t) From the Action ☒ menu select Gripper Commands, the Gripper Com... ☒ menu will appear. From this menu box select Open Hand . The gripper will now open. The actions have now been recorded.

u) Select the command Move Relative. The "Relative" dialogue box will appear, enter the values X = 0, Y = 0, Z = 100, select OK to accept. The robot will move vertical to clear both the object PART and pallet. The actions have now been recorded.

v) Using the mouse ↖, select the GP labeled "HOMEGP". Select the command GP Move. Select OK to accept. The robot will move to "HOMEGP". The actions have now been recorded.

w) Using the mouse ↖, select the object labeled " TRAYSTACKCLR". Select the command GP Move. Select OK to accept. The robot will move to " TRAYSTACKCLR". The actions have now been recorded.

x) Select the command Move Relative. The "Relative" dialogue box will appear, enter the values X = 0, Y = 0, Z = 100, select OK to accept. The robot will move vertical to clear both the object PART and pallet. The actions have now been recorded.

y) Using the mouse ↖, select the GP labeled " PALLETCLRGP". Select the command GP Move. Select OK to accept. The robot will move to " PALLETCLRGP". The actions have now been recorded.

z) Using the mouse ↖, select the GP labeled " TRAYSTACKCLR". Select the command GP Move. Select OK to accept. The robot will move to " TRAYSTACKCLR". The actions have now been recorded.

aa) Select the command Move Relative. The "Relative" dialogue box will appear, enter the values X = 0, Y = 0, Z = -225, select OK to accept. The robot will move vertical towards TRAY1. The actions have now been recorded.

bb) Select the Object "TRAY1" and From the Action ☒ menu select Gripper Commands, the Gripper Com... ☒ menu will appear. From this menu box select Close Hand, a Confirm ☒ menu box will appear, select Yes. The gripper will now close and the object "TRAY1" will be attached to the robot face plate. The actions have now been recorded.

cc)

dd) Select the command Move Away. The "Move Away" dialogue box will appear, enter the values X = 0, Y = 0, Z = 225, select OK to accept. The robot will move vertical to clear tray stack. The actions have now been recorded.

ee) Select the command Move Relative. The "Relative" dialogue box will appear, enter the values X = 350, Y = 0, Z = 0, select OK to accept. The robot will move horizontal to the loading area of the staging fixture. The actions have now been recorded.

ff) Select the command Move Relative. The "Relative" dialogue box will appear, enter the values X = 0, Y = 0, Z = -275, select OK to accept. The robot will move vertical to the loading area of the staging nest. The actions have now been recorded.

gg) From the Action ☒ menu select Gripper Commands, the Gripper Com... ☒ menu will appear. From this menu box select Open Hand . The gripper will now open and detach the object "TRAY!" from the robot face plate. The actions have now been recorded.

hh) Select the command Move Away. The "Move Away" dialogue box will appear, enter the values X = 0, Y = 0, Z = 75, select OK to accept. The robot will move vertical to clear the tray. The actions have now been recorded.

ii) Using the mouse ↖, select the GP labeled " HOMEGP ". Select the command GP Move. Select OK to accept. The robot will move to " HOMEGP ". The actions have now been recorded.

jj) From the Action ☒ menu box select End to end recording.

kk) Using the mouse ↖ select ↻ found on the left side of the tool bar. The reloading of the model may require a confirmation select Yes to confirm. Note, the track recording will not be lost.

ll) Save the project model 💾.

mm) The basic programming has been recorded and going forward all remaining programming will be completed using the program editor.

nn) Open the text editor to view the program track Robot.KL. Using the mouse ↖, select the "Robot.KL" twice in rapid successions to open the Robot.KL track in the program editor.

oo) Insert the following text to declare needed variables. The text needs to be inserted between the program description comment and the beginning of the main program. Use the recorded text to aid in editing the track.

```
CONST
  PARTSTOTAL=27

VAR
  PartsCount : INTEGER
  ROW : INTEGER
  COLUMN : INTEGER
  TRAYNO : INTEGER
  FULLTRAY: BOOLEAN

  --! SIGNALDEF DOUT[1],TRACK     -- Robot Clear Press
  --! SIGNALDEF DIN[1],Press.KL,1  -- Press Requests Part unload
```

- The constant PartsTotal will control the total number of times the track Robot.KL will loop. In this simulation the track will loop 27 times.
- The integer ROW will be indexed and used to calculate the relative move for the object's row position on the pallet.
- The integer COLUMN will be indexed and used to calculate the relative move for the object's column position on the pallet.
- The use of ROW and COLUMN is similar to a typical spread sheet.

- The integer PartsCount is a counter and is indexed per loop the program is executed.
- There are two signal definitions required to control the flow of each track as it relates to each other. The comments which follows the declarations describes the statement.

pp) Insert the following text. This is the first of three routines required. The text needs to be inserted following the variable declarations and before the main program.

```
ROUTINE PressUnload
Begin
  -- Wait for Press unload request
  WAIT FOR DIN[1]=ON
  $TERMTYPE=FINE
  $MOTYPE=LINEAR
  MOVE RELATIVE VEC(0,200,0)
  -- Use COPYOBJECTALT to copy PRESSPART
  -- This allows control over the naming of the new part
  -- COPYOBJECTALT 'name of existing object','name of new object'
  --! COPYOBJECTALALT 'PRESSPART','PART'
  --! MakeInvisible 1,'PRESSPART'
  CLOSE HAND 1
  -- Grasp object PART with the prefix NOPARENT.
  -- When multiple object having the same name exist
  -- it is difficult for WSLT to know which object with the same name to grasp.
  -- If there are more that one unattached objects, the function grasp will not
  -- occur.
  -- If there are multiple object with the same name and NOPARENT is not used,
  -- WSLT will flag this as an error and the track will fault out.
  -- ! GRASPOBJ 'NOPARENT.PART'
  -- Move clear of ejector pins
  MOVE RELATIVE VEC(0,0,10)
  -- Move clear of press
  MOVE AWAY 200.00
END PressUnload
```

qq) Insert the following text. This is the second of three required routines.

```
ROUTINE PalletLoad
Begin
  $TERMTYPE=FINE
  $MOTYPE=LINEAR
  -- Robot move to pounce position above the column row nest,
  -- The spacing between each column and row is 100mm
  -- () must be used when calculating a position.
  MOVE RELATIVE VEC((300-COLUMN*100),(-300+ROW*100),0)
  -- Robot  move to object PART to nest
  MOVE RELATIVE VEC(0,0,(-325.5+(TRAYNO*75.5)))
  OPEN HAND 1
  -- use ATTACHOBJALT to attach object PART to the object PALLET.
  -- This function allows the use of NOPARENT.
  -- NOPARENT. allows multiple object with the same name to be used.
  -- When multiple object having the same name exist
  -- it is difficult for WSLT to know which object with the same name to attach.
  -- If there are more that one unattached objects, the function used to attach the
  -- objects will not occur.
  -- If there are multiple object with the same name and NOPARENT is not used,
  -- WSLT will flag this as an error and the track will fault out.
  IF TRAYNO=1 THEN
    -- ! ATTACHOBJALT 'TRAY1','NOPARENT.PART'
  ENDIF
  IF TRAYNO=2 THEN
    -- ! ATTACHOBJALT 'TRAY2','NOPARENT.PART'
  ENDIF
  IF TRAYNO=3 THEN
    -- ! ATTACHOBJALT 'TRAY3','NOPARENT.PART'
  ENDIF
  -- Robot move clear of nest
  MOVE RELATIVE VEC(0,0,75)
  COLUMN=COLUMN+1
  IF COLUMN=4 THEN
    ROW=ROW+1
    COLUMN=1
    IF ROW=4 THEN
      FULLTRAY=TRUE
      ROW=1
    ENDIF
  ENDIF
END Palletload
```

rr) Insert the following text. This is the third of three required routines.

```
ROUTINE GETEMPTYTRAY
BEGIN
  $TERMTYPE=FINE
  $MOTYPE=JOINT
  MOVE TO PALLETCLRGP
  MOVE TO TRAYSTACKCLR
  $MOTYPE=LINEAR
  TRAYNO=TRAYNO+1
  -- Move to Grasp Tray
  MOVE RELATIVE VEC(0,0,(-200-(25*TRAYNO)))
```

```
    CLOSE HAND 1
    IF TRAYNO=1 THEN
      -- ! GRASPOBJ 'TRAY1'
      -- Move clear of tray stack
      -- Move to Tray release
    ENDIF
    IF TRAYNO=2 THEN
      -- ! GRASPOBJ 'TRAY2'
    ENDIF
    IF TRAYNO=3 THEN
      -- ! GRASPOBJ 'TRAY3'
    ENDIF
    -- Move clear of tray stack
    MOVE AWAY (200+(25.00*TRAYNO))
    -- Move to fixture tray loading
    MOVE RELATIVE VEC(350,0,0)
    -- Move to Tray release
    MOVE RELATIVE VEC(0,0,(-350.5+(75.5*TRAYNO)))
    Release tray
    OPEN HAND 1
    MOVE AWAY 75.00
    FULLTRAY=FALSE
  END GETEMPTYTRAY
```

ss) Insert the following text. This is the main portion of the track program.

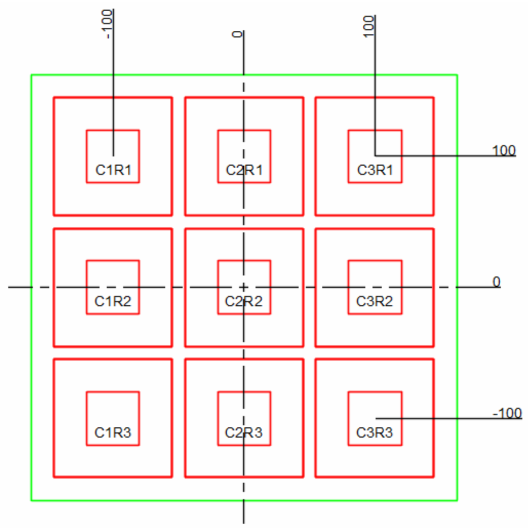
```
-- ************ Main Program ************

BEGIN
  $USEMAXACCEL=TRUE
  %INCLUDE Robot#
  $MOTYPE=JOINT
  $TERMTYPE=FINE
  $UTOOL = POS(45,0,134,0,-90,0,")
  -- Set output 1 to OFF
  DOUT[1]=OFF
  -- Set parts count to 0
  PARTSCOUNT=0
  -- SET ROW TO 1
  ROW=1
  -- SET COLUMN TO 1
  COLUMN=1
  -- Set FULLTRAY to request the first tray
  FULLTRAY=TRUE
  -- Set tray number to 1
  TRAYNO=0
  MOVE TO HOMEGP
REPEAT
  -- Retrieve an empty tray
  IF FULLTRAY=TRUE THEN
    -- Load an empty tray
    GETEMPTYTRAY
    FULLTRAY=FALSE
  ENDIF
  $MOTYPE=JOINT
  $TERMTYPE=NODECEL
  MOVE TO HOMEGP
  $TERMTYPE=FINE
  $MOTYPE=JOINT
  MOVE TO PRESSCLRGP
  PressUnload
  -- Robot cleared press and part removed
  -- Output signal ON, Read by track Press
  DOUT[1]=ON
  -- Delay output reset for .5 secs
  -- to prevent the track Press from missing the signal
  DELAY 500
  DOUT[1]=OFF
  $MOTYPE=JOINT
  $TERMTYPE=NODECEL
  MOVE TO HOMEGP
  -- Robot move to pallet clear position
  $TERMTYPE=FINE
  MOVE TO PALLETCLRGP
  PalletLoad
  PARTSCOUNT=PARTSCOUNT+1
  -- Rest PARTSCOUNT is not required, simulation will end
  -- When PARTSCOUNT=PARTSTOTAL
  IF FULLTRAY=FALSE THEN
    $MOTYPE=JOINT
    $TERMTYPE=NODECEL
    MOVE TO HOMEGP
  ENDIF
  IF PARTSCOUNT=PARTSTOTAL THEN
    $MOTYPE=JOINT
    $TERMTYPE=FINE
    MOVE TO HOMEGP
  ENDIF
  -- UNTIL is similar to a DO WHILE condition.
UNTIL PARTSCOUNT=PARTSTOTAL
END Robot
```

tt) The program track "Robot.KL" is listed in it's entirety. Comments are listed and lines are correctly indexed.

```
PROGRAM Robot
--  Workspace LT KAREL 2 Program for ABB_1200_5_90 Robot

CONST
  PARTSTOTAL=27

VAR
  PartsCount : INTEGER
  ROW : INTEGER
  COLUMN : INTEGER
  TRAYNO : INTEGER
  FULLTRAY: BOOLEAN

  --! SIGNALDEF DOUT[1],TRACK     -- Robot Clear Press
  --! SIGNALDEF DIN[1],Press.KL,1  -- Press Requests Part unload

ROUTINE PressUnload
Begin
  -- Wait for Press unload request
  WAIT FOR DIN[1]=ON
  $TERMTYPE=FINE
  $MOTYPE=LINEAR
  MOVE RELATIVE VEC(0,200,0)
  -- Use COPYOBJECTALT to copy PRESSPART
  -- This allows control over the naming of the new part
  -- COPYOBJECTALT 'name of existing object','name of new object'
  --! COPYOBJECTALT 'PRESSPART','PART'
  --! MakeInvisible 1,'PRESSPART'
  CLOSE HAND 1
  -- Grasp object PART with the prefix NOPARENT.
  -- When multiple object having the same name exist
  -- it is difficult for WSLT to know which object with the same name to grasp.
  -- If there are more that one unattached objects, the function grasp will not
occur.
  -- If there are multiple object with the same name and NOPARENT is not used,
WSLT will flag
  -- this as an error and the track will fault out.
  -- ! GRASPOBJ 'NOPARENT.PART'
  -- Move clear of ejector pins
  MOVE RELATIVE VEC(0,0,10)
  -- Move clear of press
  MOVE AWAY 200.00
END PressUnload

ROUTINE PalletLoad
Begin
  $TERMTYPE=FINE
  $MOTYPE=LINEAR
  -- Robot move to pounce position above the column row nest,
  -- The spacing between each column and row is 100mm
  -- () must be used when calculating a position.
  MOVE RELATIVE VEC((300-COLUMN*100),(-300+ROW*100),0)
  -- Robot  move to object PART to nest
  MOVE RELATIVE VEC(0,0,(-325.5+(TRAYNO*75.5)))
  OPEN HAND 1
  -- use ATTACHOBJALT to attach object PART to the object PALLET.
  -- This function allows the use of NOPARENT.
  -- NOPARENT. allows multiple object with the same name to be used.
  -- When multiple object having the same name exist
  -- it is difficult for WSLT to know which object with the same name to attach.
  -- If there are more that one unattached objects, the function used to attach the
objects will  not occur.
  -- If there are multiple object with the same name and NOPARENT is not used,
WSLT will flag
  -- this as an error and the track will fault out.  IF TRAYNO=1 THEN
  IF TRAYNO=1 THEN
    -- ! ATTACHOBJALT 'TRAY1','NOPARENT.PART'
  ENDIF
  IF TRAYNO=2 THEN
    -- ! ATTACHOBJALT 'TRAY2','NOPARENT.PART'
  ENDIF
  IF TRAYNO=3 THEN
    -- ! ATTACHOBJALT 'TRAY3','NOPARENT.PART'
  ENDIF
  -- Robot move clear of nest
  MOVE RELATIVE VEC(0,0,75)
  COLUMN=COLUMN+1
  IF COLUMN=4 THEN
    ROW=ROW+1
    COLUMN=1
    IF ROW=4 THEN
      FULLTRAY=TRUE
      ROW=1
    ENDIF
  ENDIF
END Palletload
```

```
ROUTINE GETEMPTYTRAY
BEGIN
  $TERMTYPE=FINE
  $MOTYPE=JOINT
  MOVE TO PALLETCLRGP
  MOVE TO TRAYSTACKCLR
  $MOTYPE=LINEAR
  TRAYNO=TRAYNO+1
  -- Move to Grasp Tray
  MOVE RELATIVE VEC(0,0,(-200-(25*TRAYNO)))
  CLOSE HAND 1
  IF TRAYNO=1 THEN
    -- ! GRASPOBJ 'TRAY1'
    -- Move clear of tray stack
    -- Move to Tray release
  ENDIF
  IF TRAYNO=2 THEN
    -- ! GRASPOBJ 'TRAY2'
  ENDIF
  IF TRAYNO=3 THEN
    -- ! GRASPOBJ 'TRAY3'
  ENDIF
  -- Move clear of tray stack
  MOVE AWAY (200+(25.00*TRAYNO))
  -- Move to fixture tray loading
  MOVE RELATIVE VEC(350,0,0)
  -- Move to Tray release
  MOVE RELATIVE VEC(0,0,(-350.5+(75.5*TRAYNO)))
  Release tray
  OPEN HAND 1
  MOVE AWAY 75.00
  FULLTRAY=FALSE
END GETEMPTYTRAY


-- ************  Main Program  ************

BEGIN
  $USEMAXACCEL=TRUE
  %INCLUDE Robot#
  $MOTYPE=JOINT
  $TERMTYPE=FINE
  $UTOOL = POS(45,0,134,0,-90,0,")
  -- Set output 1 to OFF
  DOUT[1]=OFF
  -- Set parts count to 0
  PARTSCOUNT=0
  -- SET ROW TO 1
  ROW=1
  -- SET COLUMN TO 1
  COLUMN=1
  -- Set FULLTRAY to request the first tray
  FULLTRAY=TRUE
  -- Set tray number to 1
  TRAYNO=0
  MOVE TO HOMEGP
REPEAT
  -- Retrieve an empty tray
  IF FULLTRAY=TRUE THEN
    -- Load an empty tray
    GETEMPTYTRAY
    FULLTRAY=FALSE
  ENDIF
  $MOTYPE=JOINT
  $TERMTYPE=NODECEL
  MOVE TO HOMEGP
  $TERMTYPE=FINE
  $MOTYPE=JOINT
  MOVE TO PRESSCLRGP
  PressUnload
  -- Robot cleared press and part removed
  -- Output signal ON, Read by track Press
  DOUT[1]=ON
  -- Delay output reset for .5 secs
  -- to prevent the track Press from missing the signal
  DELAY 500
  DOUT[1]=OFF
  $MOTYPE=JOINT
  $TERMTYPE=NODECEL
  MOVE TO HOMEGP
  -- Robot move to pallet clear position
  $TERMTYPE=FINE
  MOVE TO PALLETCLRGP
  PalletLoad
  PARTSCOUNT=PARTSCOUNT+1
  -- Rest PARTSCOUNT is not required, simulation will end
  -- When PARTSCOUNT=PARTSTOTAL
  IF FULLTRAY=FALSE THEN
    $MOTYPE=JOINT
    $TERMTYPE=NODECEL
    MOVE TO HOMEGP
  ENDIF
```

```
    IF PARTSCOUNT=PARTSTOTAL THEN
      $MOTYPE=JOINT
      $TERMTYPE=FINE
      MOVE TO HOMEGP
    ENDIF
    -- UNTIL is similar to a DO WHILE condition.
  UNTIL PARTSCOUNT=PARTSTOTAL
  END Robot
```



Fixture 1 with Two Dimensional Matrix

uu) Save the project model 💾.

2) Procedure: Create the Press track.

a) Add a track for robot MACHINEMOLD. Enter the Track Name "Press" and select the Language "KAREL 2". Using the mouse ↖ select OK when completed.

b) Using the mouse ↖, Set as Active Mechanism.

c) Open the pendent for MACHINEMOLD.

d) Using the mouse ↖, set the track program Press.KL ✓ Active.

e) Using the mouse ↖, open the text editor to view teach points. View TPs Mechanisms do not use GP's.

f) Enter the following text to record necessary teach points of the press.

```
PRESS_UP[1] = 0
-- END PRESS_UP
PRESS_DWN[1] = -88.9
-- END PRESS_DWN
PINS_DWN[2] = 0
-- END PINS_DWN
```

```
PINS_UP[2] = 15
-- END PINS_UP
```

g) Close editor once all teach points have been entered into the editor.

h) Save the project model 💾.

i) Open the text editor to view the program track Press.KL. Using the mouse ↖, select the "Press.KL" twice in rapid successions to open the Press.KL track in the program editor.

j) Insert the following text to declare needed variables. The text needs to be inserted between the program description comment and the beginning of the main program. Use the recorded text to aid in editing the track.

```
CONST
  PartsTotal=27

VAR
  PartsCount : INTEGER

  PRESS_UP : AUXPOS
  PRESS_DWN : AUXPOS
  PINS_UP : AUXPOS
  PINS_DWN : AUXPOS

  --! SIGNALDEF DOUT[1],TRACK     -- request robot unload part
  --! SIGNALDEF DIN[1],Robot.KL,1  -- Robot clear press
```

- The constant PartsTotal will control the total number of times the track Press.KL will loop. In this simulation the track will loop 9 times.
- The integer PartsCount is a counter and is indexed per loop the program is executed.
- There are 4 teach points, each declared as an "AUXPOS".
- There are two signal definitions required to control the flow of each track as it relates to each other. The comments which follows the declarations describes the statement.

k) Insert the following text. This is the main portion of the track program. there are no routines needed for this track program.

```
BEGIN
  $USEMAXACCEL=TRUE
  %INCLUDE Press#
  -- Set motion type to joint
  $MOTYPE=JOINT
  -- number of parts produced
  PartsCount=0
  -- Set all outputs to off
  DOUT[1]=OFF
  -- Set parts counter to 0
```

```
   PartsCount=0
   --! MakeInvisible 1,'PRESSPART'

 REPEAT
   -- Press to bottom stroke
   MOVE AUX TO PRESS_DWN
   -- Create new part in mold
   Delay 5000
   --! MakeVisible 1,'PRESSPART'
   -- Press raises to top stroke
   MOVE AUX TO PRESS_UP
   -- raise the ejector pins to strip part from mold cavity
   MOVE AUX TO PINS_UP
   -- Request robot unload
   DOUT[1]=on
   -- Wait for robot to unload part and clear of press
   WAIT FOR DIN[1]=ON
   DOUT[1]=OFF
   -- Once part is removed lower ejector pins
   MOVE AUX TO PINS_DWN
   -- Increment parts count
   PartsCount=PartsCount+1
 UNTIL PartsCount=PartsTotal
 END Press
```

l) Save the project model 💾.

m) Verify both program tacks Robot.KL and Press.KL are active (☑ LR MATE 200iD, ☒ Press.KL).

n) Using the mouse ↖ select ▶ (Play simulation) and observe the simulation.

o) After the simulation is finished the project must be reloaded to reset before editing or playing the simulation again. Using the mouse ↖ select ↻ found on the left side of the tool bar. The reloading of the model may require a confirmation be confirmed Yes.

3) Procedure: Create an AVI

a) Using the mouse ↖ select Simulate found on the tool bar. This will open a drop down menu.

b) from the drop down menu, using the mouse ↖ select Run Simulation and Create Animation.

c) The "Rename" dialogue box will appear, using the mouse ↖ select Open to accept the default values. The "Video Compression" menu box will open.

d) Using the mouse ↖ change the "Compressor" field to Microsoft Video 1 ▾.

e) Using the mouse ↖ select OK to begin the process of replaying the simulation and creating an AVI.

f) After the simulation is finished and the AVI has been recorded, reloaded the project model. Using the mouse ↖ select ↻ found on the left side of the tool bar.

g) The AVI can now be replayed in any Windows compatible media player program.