

EE/ME 429/529 Mechanics & Control of Robotic Manipulators

Dr. Bob and Jesus Pagan, Fall 2008

Robotics Laboratory Mini-Projects (LMP 1 – 5)

Laboratory Exercises: There will be five laboratory exercises, done in teams of two during class time in Stocker 015B (Robotics/Mechatronics/Haptics Laboratory) according to the schedule shown. Each laboratory requires a written technical report with **Memo** to be turned in at the beginning of class according to the schedule shown. Each LMP report is equal to one mini-project grade.

NO LATE ASSIGNMENTS WILL BE ACCEPTED! NO LMP GRADE WILL BE DROPPED!!

Lab work must be witnessed by our intrepid assistant Jesus Pagan and/or Dr. Bob. Your laboratory reports must include plenty of graphics – sketches, photographs, etc. All five LMPs focus on the Adept 4-dof SCARA industrial robot in Stocker 015B (see photo below).

Safety: For all hardware experiments, safety is of **PARAMOUNT IMPORTANCE!!** You must have at least one other person in the laboratory with you at all times. Use common sense and mind the **big red power kill button** (on the teach pendant and the panel) – hit it if there is an error or problem, to protect humans and the equipment. Under no circumstances are you to enter the robot workspace while it is powered up. **DO NOT** get comfortable with this robot, it can injure you badly, it can even kill!

Laboratory Mini-Projects Summary:

0. Adept SCARA Industrial Robot Demo (no report due).
1. Adept SCARA Industrial Robot orientation, modeling, and understanding.
2. File management and programming environment; DH parameters
3. Basic programming; peg-in-hole palletizing with all poses taught
4. More programming; peg-in-hole palletizing based on one pose
5. Program the robot to write a slogan or draw logo/artwork.



Adept 550 SCARA Robot

Table of Contents

LABORATORY PARTNER GROUPS.....	3
LABORATORY SCHEDULE.....	3
LABORATORY MINI-PROJECT ASSIGNMENTS.....	4
LABORATORY MINI-PROJECT LMP 0.....	4
LABORATORY MINI-PROJECT LMP 1.....	5
LABORATORY MINI-PROJECT LMP 2.....	6
LABORATORY MINI-PROJECT LMP 3.....	9
LABORATORY MINI-PROJECT LMP 4.....	10
LABORATORY MINI-PROJECT LMP 5.....	11
APPENDIX A. ADEPT 550 INDUSTRIAL SCARA ROBOT PROCEDURES.....	12
APPENDIX B. USING MOTIONWARE FOR PICK-AND-PLACE MOTIONS	14
APPENDIX C. FILE MANAGEMENT AND PROGRAMMING ENVIRONMENT	15
FILE MANAGEMENT	15
PROGRAMMING ENVIRONMENT	17
APPENDIX D. PROGRAMS FOR ROBOTIC PALLETIZING (LMP #3).....	22
APPENDIX E. PROGRAMS FOR ADVANCED ROBOTIC PALLETIZING (LMP #4)	28
APPENDIX F. FORMAT FOR LABORATORY REPORTS.....	37

Laboratory Partner Groups

1. Bryan Crosby / Ryan Wagner
2. Jon Robe / Andy Tompkins
3. Brent Morris / Kyle Sink
4. Rick Bond / Jun Ding
5. Mike McGregor / Joe Schultheis

Laboratory Schedule

Day	Lab	Groups	Due
Friday 9/12	0	1, 2, 3, 4, 5	NA
Thursday 9/25	1	1, 2, 3	Thursday 10/2
Friday 9/26	1	4, 5	Thursday 10/2
Thursday 10/2	2	1, 2, 3	Thursday 10/9
Friday 10/3	2	4, 5	Thursday 10/9
Friday 10/10	3	2, 3	Thursday 10/23
Friday 10/17	3	1, 4, 5	Thursday 10/23
Thursday 10/23	4	1, 2, 3	Thursday 10/30
Friday 10/24	4	4, 5	Thursday 10/30
Friday 10/31	5	2, 3	Thursday 11/13
Friday 11/7	5	1, 4, 5	Thursday 11/13

Laboratory Mini-Project Assignments

Laboratory Mini-Project LMP 0

Adept 550 SCARA robot demo

No report is due for this activity.

ADEPT 550 SCARA ROBOT DEMO

1. Plug in the 24V power supply and open the air supply valve.
2. Turn the controller power **on** by switching the SYSTEM POWER from **0** to **1**. (The controller will boot now)
3. **WARNING!** The Robot will move during the calibration procedure. Make sure the working envelope of the robot is clear.
4. Wait until you see **wait.start** on the terminal window and press the PROGRAM START button. (You should see the **en po** command on the terminal window indicating that the power is enable. You should also see the **cal** command which takes care of the calibration of the robot arm.)
5. After calibration, the demo files will be loaded from the floppy drive a:\.
6. Continue by following the instruction from the terminal window. When you see a black square on the screen, the system is waiting for an operator input. Go ahead and answer the question followed by a **RETURN**. If no questions are being asked, then press **RETURN** to continue.
7. Good Luck! And enjoy the demo.
8. When you are done, please turn off the SYSTEM POWER back to **0**, unplug the 24V power supply and close the air supply valve.

Laboratory Mini-Project LMP 1

Modeling/understanding/teach pendant control of the Adept 550 SCARA robot

1. Identify all active joints of the Adept 550 SCARA robot; include joint variable names for each.
2. What does SCARA stand for?
3. Measure all important dimensions for the Adept 550 SCARA robot (units: *m* and *deg*)
4. Identify the power source, actuators, and transmissions.
5. Identify the power source and actuation of the gripper.
6. Draw/sketch the robot to approximate scale. Show the motion of all joints.
7. Fire up the robot and calibrate it.
8. Exercise all functions for joint and Cartesian (world and tool coordinates) motions via the teach pendant.
9. Draw coordinate frames for the world and tool Cartesian coordinate frames. Label the world frame on the robot table (use masking tape).
10. Estimate the robot joint limits using joint control mode with the teach pendant.
11. Sketch the workspace for the Adept 550 SCARA robot (see manuals).

Laboratory Mini-Project LMP 2

DH Parameters, file management, and programming environment of the Adept 550 SCARA robot

1. Sketch the kinematic diagram, attach the coordinate frames, and derive the **Denavit-Hartenberg Parameters** for the Adept 550 SCARA robot. This can be done out of the lab – just ensure you understand the robot well enough to sketch the kinematic diagram and derive the DH parameters.

2. File Management

(Insert the File Management Disk into the floppy disk drive)

The File Management Disk should have a single file named one.v2 which contains three programs named one, two and three. This file also contains various location points stored in it.

1. Perform a system start-up.
2. Display the time. (**time**)
3. Set the default data disk to the a:\ disk drive. (**cd a:** or **cd a:**)
4. Obtain a File Directory of the contents of the a:\ disk drive. (**fdir**)
5. Obtain a File Directory of the contents of the c:\ disk drive. (**fdir c:**)
Note: (Your current disk drive is still a:\)
6. Obtain a list of all program names in system memory. (**dir**)
7. Load the file named one.v2 from the a:\ disk drive into memory. (**load one.v2**)
8. Obtain a list of all program names in system memory.
9. Try loading the file named one.v2 again and see what happens.
10. Take a look at the programs that were loaded into system memory. (**listp**)
11. Take a look at the location points that were loaded into memory. (**listl**)
12. Store the programs named one and two into a file named “myfile”.
(**store myfile=one,two**)
Note: (a file myfile.v2 was created in the active disk drive which should contain the two programs as well as all location points and variables, if any, that were in system memory.)
13. Obtain a File Directory of the contents of the a:\ disk drive.
14. Store the programs named two and tree into a file named “myfile” and see what happens.
Note: (files are not replaced if the file name already exists)
15. Store just the program named one in a file named one. (**storep one=one**)

16. Store the location points in system memory to a file named one. (**storel one**)
17. Obtain a File Directory of the contents of the a:\ disk drive.
Note: (notice the extension of the files named one.
.PG for programs
.LC for location variables
.V2 for a combination of programs, locations and variables)
18. Rename the disk file “myfile.v2” to the name “file_21.v2”. (**fren file_21.v2=myfile.v2**)
19. Obtain a File Directory of the contents of the a:\ disk drive.
Note: (the old file “myfile.v2” does not exists anymore)
20. Copy the disk file “file_21” to a new disk file “file_12”. (**fcopy file_12.v2=file_21.v2**)
21. Obtain a File Directory of the contents of the a:\ disk drive.
22. List the contents of disk file “file_12”. (**flist file_12.v2**)
23. List the contents of disk file “file_21”.
24. Obtain a File Directory of the contents of the a:\ disk drive.
25. Delete the disk file “file_21”. (**fdel file_21.v2**)
Note: (Are you sure (Y/N)? enter a “y” and press ENTER)
26. Set the default data disk to the c:\ disk drive.
27. Obtain a File Directory of the contents of the c:\ disk drive.
28. Set the default data disk to the a:\ disk drive.
29. Obtain a File Directory of the contents of the a:\ disk drive.
30. Create a subdirectory on the a:\ disk drive named “oakid”. (**fdir/c oakid**)
31. Obtain a File Directory of the contents of the a:\ disk drive.
Note: (you are still in the a:\ disk drive and not in the newly created subdirectory)
32. Copy the disk file “file_12.v2” from the a:\ disk drive to the newly created subdirectory in the a:\ disk drive using the same file name.
(**fcopy a:\oakid\file_12.v2=file_12.v2**)
33. Obtain a File Directory of the contents of the “oakid” subdirectory in a:\ disk drive.
34. Rename the disk file “file_12” in the subdirectory “oakid” to file “file_3.v2” and see what happens. (**fren a:\oakid\file_3.v2=a:\oakid\file_12.v2**)

35. Set the default data disk to the a:\oakid\ disk drive subdirectory.
36. Obtain a File Directory of the contents of the “oakid” subdirectory in a:\ disk drive.
37. Rename the disk file “file_12” in the subdirectory “oakid” to file “file_3.v2”.
38. Obtain a File Directory of the contents of the “oakid” subdirectory in a:\ disk drive.
39. Set the default data disk to the a:\ disk drive.
40. Obtain a File Directory of the contents of the “oakid” subdirectory in a:\ disk drive.
41. Delete the disk file “file_3.v2” from the “oakid” subdirectory on the a:\ disk drive.
42. Obtain a File Directory of the contents of the “oakid” subdirectory in a:\ disk drive.
43. Delete the “oakid” subdirectory from the a:\ disk drive. (**fdir/d oakid**)
44. Obtain a File Directory of the contents of the a:\ disk drive.
45. Delete the file “one.pg”.
46. Delete the file “one.lc”.
47. Delete the file “file_12.v2”.
48. Obtain a File Directory of the contents of the a:\ disk drive.
49. Zero the system memory. (**zero**)
50. Obtain a list of all program names in system memory.

Laboratory Mini-Project LMP 3

Basic programming; peg-in-hole palletizing with all poses (positions and orientations) taught

Write a program to perform robotic palletizing with the Adept 550 SCARA robot. Specifically, transfer all cylindrical pegs from the 3x3 source pallet (pick) on the robot's right to the 3x3 destination pallet (place) on the robot's left. Use the 18 pre-taught points that Jesus will hook you up with.

This lab is to be performed by each team separately, so we will need to write and edit programs off-line using a text editor on a lab PC. Take turns with the robot to test, modify, and demonstrate your palletizing program.

See Appendix D for a sample program developed by Jesus to aid in your programming.

Laboratory Mini-Project LMP 4

More programming; peg-in-hole palletizing based on one pose

Write a program to again perform robotic palletizing with the Adept 550 SCARA robot. Specifically, transfer all cylindrical pegs from the 3x3 source pallet (pick) on the robot's right to the 3x3 destination pallet (place) on the robot's left.

This time use only one pre-taught point on the 3x3 source pallet and one pre-taught point on the 3x3 destination pallet. Use the commands **shift** and **transformation** to accomplish your palletizing task. These will tell the robot to move relative to your anchor point on each pallet.

This lab is again to be performed by each team separately, so we will need to write and edit programs off-line using a text editor on a lab PC. Take turns with the robot to test, modify, and demonstrate your updated palletizing program.

Laboratory Mini-Project LMP 5

Program the robot to write a slogan or draw logo/artwork with dry-erase markers

Using what you have learned in the first programming labs, write a program to make the Adept 550 SCARA robot write and/or draw something of your team's own design using dry erase markers on 8.5x11" paper.

CAUTION: your robot artwork must be family-friendly only!

Specifically, design your idea in advance and calculate the points for the robot to re-create off-line to save time on lab morning. You must use a minimum of two colors, and you can use up to 4 colors: black, red, blue, green.

Demonstrate your robot artwork in lab to Jesus and Dr. Bob and then submit the final drawing/writing as part of your lab #5 report.

Jesus will prepare eight programs for your use:

```
pick.peg.black
pick.peg.red
pick.peg.green
pick.peg.blue

return.peg.black
return.peg.red
return.peg.green
return.peg.blue
```

to pick up the desired colored pen and then to return it after use.

This lab is again to be performed by each team separately, so you will need to write and edit programs off-line using a text editor on a lab PC. Take turns with the robot to test, modify, and demonstrate your drawing/writing program. You should develop your basic points for your work at home prior to coming to the lab on your scheduled morning.

See **PROGRAM arc** at the end of Appendix E for an example of generating a circle for drawing.

You and your partner are welcome to come to the lab early (tell Dr. Bob and/or Jesus), stay late, or come for another session after lab day, in order to make nice artwork. Your challenge is to make it simple enough to be do-able, but complicated enough to impress Jesus.

Appendix A. Adept 550 Industrial SCARA Robot Procedures

System Start-up

Note: Before turning on power to the equipment, make sure all cables are installed correctly.

1. SYSTEM POWER on (optional front panel).

List of components with power switches (these in general remain on):

- a. Circuit breaker.
 - b. Power Chassis (PA-4).
 - c. MV Controller (MV-19).
 - d. Terminal Screen (VGA Monitor).
2. After the boot up sequence, the dot “.” prompt, a period, will appear in the lower left corner of the terminal screen.

Calibration

(Using the System Terminal Keyboard)

1. EMERGENCY STOP switches are pulled out (MCP and optional front panel).
2. Key Switch to “TERMINAL” (optional front panel).
3. **en po** ENTER (enable power).
4. **cal** ENTER (calibrate).
5. Are you sure (y/n)? Respond with a **y** ENTER.
6. The robot will move during the calibration process.

Shut-down

1. Press the EMERGENCY STOP or **dis po** ENTER (disable power).
2. SYSTEM POWER off (optional front panel).

Below, the stuff you type is in *italics*. Pull-down menu commands and teach pendant buttons are in **bold**. Underlines are used for emphasis. The Adept MotionWare software is a trifle non-user-friendly. Use the trackball to make choices; sometimes you must double-click, and sometimes there is a significant lag so you must be patient.

Power-up Process

- Power on monitor
- Switch Controller on.
- Switch Power Chassis on.
- Turn System Power switch on Main Control Panel (MCP) to 1.
- Wait for system to boot up – A “.” Prompt appears on the monitor.
- Type *enable power* on V+ controller. HIGH POWER light on MCP will activate.
- Type *calibrate*, ensure workspace is clear and confirm *Y*. Robot will perform joint calibration.
- To use Teach Pendant, see below. To use MotionWare for pick-and-place motions (also requires the teach pendant), see flip-side.

Teach Pendant

- Verify that the teach pendant is plugged into the MCP
- Turn key on MCP to Pendant.
- Press the **CMP/PWR** button.
- Press the **MAN/HALT** button to select the type of pendant motion (**World**, **Tool**, or **Joint**).
- Select the desired Cartesian direction/joint: **X/1**, **Y/2**, **Z/3**, **RX/4**, **RY/5**, **RZ/6**. Note that Adept sells other, more general robots and not all of these motions work for our 4-dof SCARA robot.
- The Speed Bars will move the robot in one Cartesian or joint axis depending on your **MAN/HALT** choice. You can choose + or – motions, with continuous speed control from painfully slow to faster.

Power-down Process

- Make sure robot is in a good resting configuration.
- Type *disable power*.
- Turn System Power switch on MCP to 0.
- Switch Power Chassis off.
- Switch Controller off.
- Power off monitor

Automated Procedures

Appendix B. Using MotionWare for Pick-and-Place Motions

Loading and Executing MotionWare Software

At the “.” Prompt, type: *default disk = c:\mow*
 load lmow
 comm lmow

If the execution (*comm*) fails, try pressing the **COMP/PWR** pendant button and re-typing *comm lmow*.
 Non-existent file message – click **Continue**.

You are now in the Adept MotionWare window:

Adept Execute Edit I/O Show Set-up Utilities Special Help

Teaching Pick-and-Place Poses

Utilities – Locations Management – Create – type in desired *name* for your locations file

Edit – Locations – choose the *name* you created; it will tell you no records exist

Press **NEW** button (**F2**) to create your first and ensuing poses (leave key pointing to **Terminal**):

NEW (F2) – type pose name (*name1*) - use teach pendant to define new pose – click **HERE**
 Repeat for all desired poses (*name1-n*).

** Ensure all poses have zeroes in the **Approach** and **Depart** boxes; also check the **mm/s** boxes **

Creating a Motion Sequence

Utilities – Sequence Management – Create – type in desired *title* (different from *name*) for your sequence file – also be sure to enter your locations file *name* in the third field.

Edit – Sequence – choose the *name* you created; it will tell you no records exist

F2 (new) – statement – Highlight and press **F3** to get a list of valid statements; choose **move (retrieve)**

MOVE TO – location – Highlight and press **F3** to get desired pose **namei** from your locations
 Repeat for as many poses as you desire – end with **STOP_ROBOT**.

Running a Motion Sequence

First, ensure the **COMP/PWR** button is pressed on the teach pendant.

Utilities – Sequence Management – Load – double-click on your sequence *title* (skip if your sequence is already loaded).

Execute – Operator Control Panel – Index – double-click on *title*. Then click **Start**. Enjoy motion.

To Exit

Special – System Shutdown – Yes – then follow power-down process on flip-side.

Appendix C. File Management and Programming Environment

File Management

Monitor Commands – issued at the period (dot) prompt.

Disk file manipulation
Control program execution
Display system status
Return various functions

ID
STATUS
FREE
TIME dd-mm-yy hh:mm:ss

Information Management

Disk files – several items such as programs, variables, and locations

Loading	→	load work1.v2	to system memory
Storing	→	store work1.v2	to hard drive / floppy drive

The Adept 550 SCARA robot uses the commercial MotionWare software based on the V+ language (Val+); the default file extension is **v2**.

```
store file_spec = program_name1, program_name2, . . .
```

store P	programs and subprograms
store L	location variables
store R	real variables
store S	string variables

Disk File Management

```
frename path new_file = old_file  
  
fdelete path file_spec  
  
flist path file_spec  
  
fcopy path new_file_spec = path old_file_spec
```

Directories

```
fdir path list directories
```

```

fdir /c c:\example\          create the directory example
fdir /c c:\example\one\     create the subdirectory one
fdir /d c:\example\one\     delete the subdirectory one

```

Note: before you can delete a directory, all files and data within that directory must be deleted (**fdelete**).

Disk Drives

```

format a:                  erase and format floppy disk
default disk = path        establish default directory path

```

System Memory

```
rename new_program = old_program
```

```
delete X name1, name2, ...
```

```

X:  P          programs and subprograms
      L          location variables
      R          real variables
      S          string variables

```

```
zero
```

```
list X name1, name2, . . .
```

```
copy new_program = old_program
```

```
dir
```

```
dir1
```

Module – a group of programs that reside in system memory and can be referred to by a single name.

```
mdir module_name
```

```
module module_name = program_name1, program_name2, . . .
                        (the first program name should be used as the module name)
```

```
delete m
```

```
store m diskfile.ext = module_name
```

Programming Environment

Type of Programs

Robot control program	directly controls the robot
General program	monitor and control external processes using I/O
Monitor command program	start up sequences and system level tasks

Editor

see program_name

create, view, and modify programs

program program_name()

editor modes	command	special commands (initial mode, F11)
	insert	(i) (insert key)
	replace	(F12)

cursor control arrow keys

return / backspace / delete keys

cut, copy, paste

search, find, change

exit the editor (F4)

editor demo – p. 113, Section 5

Data Types

real	.number = -62	
string	.\$string = "adept"	
precision point	.set #start = #ppoint(-135,jt2,75,90)	robot joint angles
transformation	.set pick = trans(450,y_val,708,0,180,-45)	robot tool location and orientation
arrays	array_name[row]	1D
	array_name[row,column]	2D
	array_name[row,column,page]	3D

Program Instructions

operators	() * / + -
comments	;
type	output1, output2, . . . /cn clear <i>n</i> blank lines /un move the cursor up <i>n</i> lines /xn move the cursor right <i>n</i> spaces /s hold the cursor at the current location
prompt	output, variable

Program Design

Software packages – group of similar functions

Header program

Main application

Subprograms

Tasks programs

Example Program Architecture:

```

.program a.software( ) ; comments here
; descriptive comment
; version info
; header info/copyright
; description
; special instructions
; global entry points
; author:
; changes:

call swr.main( ) ; invoke main routine
return ; software end of program

; specify dummy calls
call swr.task1( )
call swr.task2( )

.end

```

```
software.v2
```

```
a.software (header)
swr.main (main)
```

```
swr.sub1
swr.sub2
```

```
swr.task1
swr.task1.sub1
```

```
task:
```

Program Control Structures

```
goto label label from 0 to 32,767
```

```
call subprogram( )
```

```
if relationship goto label
```

```
if relationship then
```

```
. . .
```

```

else
    . . .
end

case value of
    value1:
        . . .
    value2,3,4:
        . . .
    any
        . . .
end

for index = initial_value to final_value step value
    . . .
end

while condition do
    . . .
end

do
    . . .
until condition

```

Program Execution

System terminal

Before execution – load programs, locations, variables

execute a program, number_cycles, starting_step

xstep a program, number_cycles, starting_step

abort

pause

proceed

kill

retry

From the MCP (manual control panel) – programs must be in memory (**load**)

Key switch to pendant

Select program to execute

System Variables

switches – Table S-1, p. 133

parameters – Table S-2, p. 134

enable switch_name

disable switch_name

switch switch_name = value

0 – switch is off

-1 – switch is on (or any other number)

switch to see status of all system switches

Appendix D. Programs for Robotic Palletizing (LMP #3)

by Jesus Pagan, Fall 2008

```
.PROGRAM grip.off()  
    SIGNAL (2)  
    DELAY 0.1  
    SIGNAL (-2)  
    DELAY 0.1  
    RETURN
```

```
.END
```

```
.PROGRAM grip.on()  
    SIGNAL (1)  
    DELAY 0.1  
    SIGNAL (-1)  
    DELAY 0.1  
    RETURN
```

```
.END
```

```
.PROGRAM grip.reset()  
    SIGNAL (-1)  
    DELAY 0.1  
    SIGNAL (-2)  
    DELAY 0.1  
    SIGNAL (2)  
    DELAY 0.1  
    SIGNAL (-2)  
    DELAY 0.1  
    RETURN
```

```
.END
```

```
.PROGRAM p1p2()  
.END
```

```
.PROGRAM p2p1()  
.END
```

```
.PROGRAM pallet.one.pick(loc)  
  APPRO pick[loc], 50  
  BREAK  
  MOVE pick[loc]  
  BREAK  
  CALL grip.on()  
  DEPART 50  
  BREAK  
  
.END
```

```
.PROGRAM pallet.one.plac(loc)  
  APPRO pick[loc], 50  
  BREAK  
  MOVE pick[loc]  
  BREAK  
  CALL grip.off()  
  DEPART 50  
  BREAK  
  
.END
```

```
.PROGRAM pick(loc)  
  APPRO pick[loc], 50  
  BREAK  
  MOVE pick[loc]  
  BREAK  
  CALL grip.on()  
  DEPART 50  
  BREAK  
  
.END
```

```
.PROGRAM pickplace()  
  CALL grip.reset()  
  MOVE #safe  
  FOR loc = 1 TO 9 STEP 1  
    APPRO pick[loc], 50  
    BREAK  
    MOVE pick[loc]  
    BREAK  
    CALL grip.on()  
    DEPART 50  
    BREAK  
    APPRO place[loc], 50  
    BREAK  
    MOVE place[loc]  
    BREAK  
    CALL grip.off()  
    DEPART 50  
    BREAK  
  END  
  MOVE #safe
```

```
.END
```

```
.PROGRAM place(loc)  
  APPRO place[loc], 50  
  BREAK  
  MOVE place[loc]  
  BREAK  
  CALL grip.off()  
  DEPART 50  
  BREAK
```

```
.END
```

```
.PROGRAM placepick()
  CALL grip.reset()
  MOVE #safe
  FOR loc = 1 TO 9 STEP 1
    APPRO place[loc], 50
    BREAK
    MOVE place[loc]
    BREAK
    CALL grip.on()
    DEPART 50
    BREAK
    APPRO pick[loc], 50
    BREAK
    MOVE pick[loc]
    BREAK
    CALL grip.off()
    DEPART 50
    BREAK
  END
  MOVE #safe
.END
```

.LOCATIONS

```

pick[0]  0.998718858 -0.050602421 0 -0.050602421 -0.998718858 0
0 0 -1 321.985443115 -251.274078369 146.99571228
pick[1]  0.998718858 -0.050602421 0 -0.050602421 -0.998718858 0
0 0 -1 321.985443115 -251.274078369 146.99571228
pick[2]  0.998704791 -0.050879642 0 -0.050879642 -0.998704791 0
0 0 -1 322.592559814 -216.334152222 146.095932007
pick[3]  0.998704493 -0.050884794 0 -0.050884794 -0.998704493 0
0 0 -1 322.992248535 -181.770492554 145.960708618
pick[4]  0.994113684 -0.108341955 0 -0.108341955 -0.994113684 0
0 0 -1 287.710357666 -250.920730591 146.4193573
pick[5]  0.995020688 -0.099668756 0 -0.099668756 -0.995020688 0
0 0 -1 287.596191406 -215.746871948 145.790710449
pick[6]  0.998705745 -0.050861254 0 -0.050861254 -0.998705745 0
0 0 -1 288.136199951 -181.018585205 146.071304321
pick[7]  0.995023549 -0.099640101 0 -0.099640101 -0.995023549 0
0 0 -1 251.999130249 -250.275497437 145.363769531
pick[8]  0.995025396 -0.099621721 0 -0.099621721 -0.995025396 0
0 0 -1 252.805450439 -215.47644043 145.311325073
pick[9]  0.998705566 -0.050863951 0 -0.050863951 -0.998705566 0
0 0 -1 253.28692627 -180.567398071 145.685134888
place[0] 0.588077605 0.808804512 0 0.808804512 -0.588077605 0
0 0 -1 293.793579102 233.628921509 149.327682495
place[1] 0.588077605 0.808804512 0 0.808804512 -0.588077605 0
0 0 -1 293.793579102 233.628921509 149.327682495
place[2] 0.480606228 0.876936495 0 0.876936495 -0.480606228 0
0 0 -1 258.588531494 198.619613647 149.176574707
place[3] 0.505987167 0.86254096 0 0.86254096 -0.505987167 0
0 0 -1 328.266876221 196.973449707 149.587722778
place[4] 0.495715648 0.868484855 0 0.868484855 -0.495715648 0
0 0 -1 328.91973877 266.727355957 150.047622681
place[5] 0.239545241 0.970885217 0 0.970885217 -0.239545241 0
0 0 -1 259.115020752 268.270202637 148.294845581
place[6] 0.492371947 0.870384872 0 0.870384872 -0.492371947 0
0 0 -1 327.604736328 231.982833862 151.083709717
place[7] 0.365635574 0.930758119 0 0.930758119 -0.365635574 0
0 0 -1 293.770782471 267.917907715 149.478286743
place[8] 0.226237446 0.974072218 0 0.974072218 -0.226237446 0
0 0 -1 257.701690674 233.539230347 149.667098999
place[9] 0.369769394 0.92912358 0 0.92912358 -0.369769394 0
0 0 -1 291.788360596 197.51373291 151.108657837
#safe    -47.599498749 109.995605469 9.119999886 46.560001373
.END

```

```
.REALS  
  loc          10  
.END
```

```
.DOUBLES  
.END
```

```
.STRINGS  
.END
```

Appendix E. Programs for Advanced Robotic Palletizing (LMP #4)

by Jesus Pagan, Fall 2008

1. Execute the utility program **pick.peg** to pick the peg located in the **pick[1]** location.
2. Calculate the X, Y and Z distances between your assigned location and the **pick[1]** location and record them in the table provided under the **SHIFT** section.
3. Modify the **S3** program using a text editor or the **SEE** editor.
4. Execute the **S3** program to observe the robot motion. (note your observations)
5. Return the peg to it's original location by executing the **return.peg** program.
6. Explore the T2 and T3 programs.
 - a. What did you noticed?
 - b. Can you calculate the X, Y, Z, Y, P and R for two of the place[?] locations?
 - c. Try modifying the T2 and T3 programs to take into account the new transformation values so t

Gripper Programs

```
.PROGRAM grip.off()
  SIGNAL (2)
  DELAY 0.1
  SIGNAL (-2)
  DELAY 0.1
  RETURN
.END
```

```
.PROGRAM grip.on()
  SIGNAL (1)
  DELAY 0.1
  SIGNAL (-1)
  DELAY 0.1
  RETURN
.END
```

```
.PROGRAM grip.reset()
  SIGNAL (-1)
  DELAY 0.1
  SIGNAL (-2)
  DELAY 0.1
  SIGNAL (2)
  DELAY 0.1
  SIGNAL (-2)
  DELAY 0.1
  RETURN
.END
```

Utility Programs

```
.PROGRAM pick.peg()  
  CALL grip.reset()  
  MOVE #safe  
  APPRO pick[1], 50  
  BREAK  
  MOVE pick[1]  
  BREAK  
  CALL grip.on()  
  BREAK  
  DEPART 50  
  BREAK  
.END  
  
.PROGRAM return.peg()  
  APPRO pick[1], 50  
  BREAK  
  MOVE pick[1]  
  BREAK  
  CALL grip.off()  
  BREAK  
  DEPART 50  
  BREAK  
  MOVE #safe  
.END
```

SHIFT

What if S1 is the pick[1] location while S2 and S3 are any other two pick[?] locations so that we can calculate the distance in the x, y and z directions between the pick[1] and the pick[?] locations. Let's now fill in the table for the assigned pick[?] location and calculate the distances between the specified locations.

Locations	X	Y	Z
S1(pick[1])	321.985443115	-251.274078369	146.99571228
S2(pick[2])	322.592559814	-216.334152222	146.095932007
S3(pick[])			
S2 to S1	0.607	34.939	-0.899
S3 to S1			

```
.PROGRAM s2()
  SET s2 = SHIFT(pick[1] BY 0.607,34.939,-0.899)
  APPRO s2, 50
  BREAK
  MOVE s2
  BREAK
  CALL grip.off()
  BREAK
  DEPART 50
  BREAK
  DELAY 5
  MOVE s2
  BREAK
  CALL grip.on()
  BREAK
  DEPART 50
  BREAK
.END
```

```
.PROGRAM s3()
  SET s3 = SHIFT(pick[1] BY 0,0,0)
  APPRO s3, 50
  BREAK
  MOVE s3
  BREAK
  CALL grip.off()
  BREAK
  DEPART 50
  BREAK
  DELAY 5
  MOVE s3
  BREAK
  CALL grip.on()
  BREAK
  DEPART 50
  BREAK
.END
```

TRANS

Locations	X	Y	Z	Y	P	R
T1(pick[1])						
T2(pick[2])						
T3(pick[])						
T2 to T1						
T3 to T1						

```
.PROGRAM t2()
  SET t2 = pick[1]:TRANS(0,0,0,0,0,0)
  APPRO t2, 50
  BREAK
  MOVE t2
  BREAK
  CALL grip.off()
  BREAK
  DEPART 50
  BREAK
  DELAY 5
  MOVE t2
  BREAK
  CALL grip.on()
  BREAK
  DEPART 50
  BREAK
.END
```

```
.PROGRAM t3()
  SET t3 = pick[1]:TRANS(0,0,0,0,0,0)
  APPRO t3, 50
  BREAK
  MOVE t3
  BREAK
  CALL grip.off()
  BREAK
  DEPART 50
  BREAK
  DELAY 5
  MOVE t3
  BREAK
  CALL grip.on()
  BREAK
  DEPART 50
  BREAK
.END
```

```

.PROGRAM motion()
; Get locations to define the pallet      (page 125 V+ Language User's Guide)
;                                         (
;                                         Version 11.0)
    SPEED 10 ALWAYS
    CALL grip.reset()
    MOVE #safe
    APPRO place[1], 50
    BREAK
    MOVE place[1]
    BREAK
    CALL grip.on()
    BREAK
    BREAK
    DEPART 50

    DETACH ()      ;Release the robot for use by the MCP
    PROMPT "Place robot at pallet origin. ", $ans
    HERE loc.origin      ;Record the frame origin

    PROMPT "Place robot at point on the pallet x axis. ", $ans
    HERE loc.x.axis      ;Record point on x-axis

    PROMPT "Place robot at point in positive y direction. ", $ans
    HERE loc.pos.y      ;Record positive y direction

    ATTACH ()      ;Re-attach the robot

    DEPART 50
    BREAK
    MOVE #safe
    BREAK
    CALL grip.off()

; Create the local reference frame "pallet.frame"

    SET pallet.frame = FRAME(loc.origin,loc.x.axis,loc.pos.y,loc.origin)

    cx = ABS(DX(loc.origin)-DX(loc.x.axis))
    cy = ABS(DY(loc.origin)-DY(loc.x.axis))
    cz = SQRT(SQR(cx)+SQR(cy))
    cell.space = cz/2

; Remove the palletized items
    a = 1
    FOR i = 0 TO 2
        FOR j = 0 TO 2
            APPRO pick[a], 50
            BREAK
            MOVE pick[a]
            BREAK
            CALL grip.on()
            DEPART 50
            BREAK
            APPRO pallet.frame:TRANS(i*cell.space,j*cell.space), 50
            BREAK
            MOVE pallet.frame:TRANS(i*cell.space,j*cell.space)
            BREAK
            CALL grip.off()
            DEPART 50
            a = a+1
        END
    END
.END

```

```

.LOCATIONS
pick[0]      0.998718858 -0.050602421 0 -0.050602421 -0.998718858 0
0 0 -1 321.985443115 -251.274078369 146.99571228
pick[1]      0.998718858 -0.050602421 0 -0.050602421 -0.998718858 0
0 0 -1 321.985443115 -251.274078369 146.99571228
pick[2]      0.998704791 -0.050879642 0 -0.050879642 -0.998704791 0
0 0 -1 322.592559814 -216.334152222 146.095932007
pick[3]      0.998704493 -0.050884794 0 -0.050884794 -0.998704493 0
0 0 -1 322.992248535 -181.770492554 145.960708618
pick[4]      0.994113684 -0.108341955 0 -0.108341955 -0.994113684 0
0 0 -1 287.710357666 -250.920730591 146.4193573
pick[5]      0.995020688 -0.099668756 0 -0.099668756 -0.995020688 0
0 0 -1 287.596191406 -215.746871948 145.790710449
pick[6]      0.998705745 -0.050861254 0 -0.050861254 -0.998705745 0
0 0 -1 288.136199951 -181.018585205 146.071304321
pick[7]      0.995023549 -0.099640101 0 -0.099640101 -0.995023549 0
0 0 -1 251.999130249 -250.275497437 145.363769531
pick[8]      0.995025396 -0.099621721 0 -0.099621721 -0.995025396 0
0 0 -1 252.805450439 -215.47644043 145.311325073
pick[9]      0.998705566 -0.050863951 0 -0.050863951 -0.998705566 0
0 0 -1 253.28692627 -180.567398071 145.685134888
place[0]     0.588077605 0.808804512 0 0.808804512 -0.588077605 0
0 0 -1 293.793579102 233.628921509 149.327682495
place[1]     0.588077605 0.808804512 0 0.808804512 -0.588077605 0
0 0 -1 293.793579102 233.628921509 149.327682495
place[2]     0.480606228 0.876936495 0 0.876936495 -0.480606228 0
0 0 -1 258.588531494 198.619613647 149.176574707
place[3]     0.505987167 0.86254096 0 0.86254096 -0.505987167 0
0 0 -1 328.266876221 196.973449707 149.587722778
place[4]     0.495715648 0.868484855 0 0.868484855 -0.495715648 0
0 0 -1 328.91973877 266.727355957 150.047622681
place[5]     0.239545241 0.970885217 0 0.970885217 -0.239545241 0
0 0 -1 259.115020752 268.270202637 148.294845581
place[6]     0.492371947 0.870384872 0 0.870384872 -0.492371947 0
0 0 -1 327.604736328 231.982833862 151.083709717
place[7]     0.365635574 0.930758119 0 0.930758119 -0.365635574 0
0 0 -1 293.770782471 267.917907715 149.478286743
place[8]     0.226237446 0.974072218 0 0.974072218 -0.226237446 0
0 0 -1 257.701690674 233.539230347 149.667098999
place[9]     0.369769394 0.92912358 0 0.92912358 -0.369769394 0
0 0 -1 291.788360596 197.51373291 151.108657837
#safe      -2.332499981 111.291603088 9.126667023 46.560001373
.END

.REALS
.END

.DOUBLES
.END

.STRINGS
.END

```

```
.PROGRAM pick.peg()  
  CALL grip.reset()  
  MOVE #safe  
  APPRO pick[1], 50  
  BREAK  
  MOVE pick[1]  
  BREAK  
  CALL grip.on()  
  BREAK  
  DEPART 50  
  BREAK  
.END
```

```
.PROGRAM s2()  
  SET s2 = SHIFT(pick[1] BY 0.607,34.939,-0.899)  
  APPRO s2, 50  
  BREAK  
  MOVE s2  
  BREAK  
  CALL grip.off()  
  BREAK  
  DEPART 50  
  BREAK  
  DELAY 5  
  MOVE s2  
  BREAK  
  CALL grip.on()  
  BREAK  
  DEPART 50  
  BREAK  
.END
```

```
.PROGRAM s3()  
  SET s3 = SHIFT(pick[1] BY 0,0,0)  
  APPRO s3, 50  
  BREAK  
  MOVE s3  
  BREAK  
  CALL grip.off()  
  BREAK  
  DEPART 50  
  BREAK  
  DELAY 5  
  MOVE s3  
  BREAK  
  CALL grip.on()  
  BREAK  
  DEPART 50  
  BREAK  
.END
```

```
.PROGRAM return.peg()
  APPRO pick[1], 50
  BREAK
  MOVE pick[1]
  BREAK
  CALL grip.off()
  BREAK
  DEPART 50
  BREAK
  MOVE #safe
.END

.PROGRAM t2()
  SET t2 = pick[1]:TRANS(0,0,0,0,0,0)
  APPRO t2, 50
  BREAK
  MOVE t2
  BREAK
  CALL grip.off()
  BREAK
  DEPART 50
  BREAK
  DELAY 5
  MOVE t2
  BREAK
  CALL grip.on()
  BREAK
  DEPART 50
  BREAK
.END

.PROGRAM t3()
  SET t3 = pick[1]:TRANS(0,0,0,0,0,0)
  APPRO t3, 50
  BREAK
  MOVE t3
  BREAK
  CALL grip.off()
  BREAK
  DEPART 50
  BREAK
  DELAY 5
  MOVE t3
  BREAK
  CALL grip.on()
  BREAK
  DEPART 50
  BREAK
.END
```

```
.PROGRAM arc(r, o, s, e)
; r = the radius of the arc
; o = center of the circle
; s = the start angle in degree of the arc
; e = the end angle in degrees of the arc
; start location will be      x = r*cos(s)
;                             y = r*sin(s)
; end location will be      x = r*cos(e)
;                             y = r*sin(e)
; travel is clockwise

    SET one = TRANS(0,0,0,0,180,0)
    SET two = TRANS(10,0,0,0,180,0)
    SET three = TRANS(0,-10,0,0,180,0)
    SET frame1 = FRAME(one,two,three,o)
    angle = s
    APPRO frame1:TRANS(r*COS(angle),r*SIN(angle),0,0,0,0), 50
    FOR angle = s TO e STEP 1
        MOVE frame1:TRANS(r*COS(angle),r*SIN(angle),0,0,0,0)
    END
    DEPART 50
.END
```

Appendix F. Format for Laboratory Reports

1. The cover sheet must be the memo, serving as the abstract and mini-results, mini-discussion, and mini-conclusion.
2. Laboratory problem statement – taken from the Laboratory Assignment for LMP i , $i = 1,2,3,4,5$.
3. Results
4. Discussion
5. Conclusion
6. References
7. Appendices (if necessary)

Be sure to use plenty of graphics, sketches, drawings, and photos to support your laboratory reports.