Application Development

UMI-R3-111

CRS *It's all about time*

Application Development

Revision		
Number	**History**	**Date**
001	First release.	99-07

Contents

CHAPTER 1

Introduction

This Guide, Application Development, describes how to develop and run a robot automation application using:

- teach pendant
- Robcomm3
- ash

For each task, one of the development tools is recommended. The task is described using the recommended tool. In many cases, an alternative tool can be used to accomplish the same task. Refer to the reference section of that tool to use it for the task.

You can use one of two methods.

Teach-First Method

The teach-first chapter describes how to develop a simple application by teaching locations first and then writing your program.

Use this method if you prefer a hands-on approach. Walk through your application, moving the actual arm in the workcell, saving the locations, and determining the order of robot work. Then write your program to follow the order of work.

Write-First Method

The write-first chapter describes how to develop a simple application by writing your program first and then teaching the locations.

Use this method if you prefer a large-scale design approach. Determine the work of the robot. Write the program, adding sections until the program until complete. Then, using the program’s location names, teach the locations with the arm.

You can use this method to get started on your application when you do not yet have the robot installed to teach locations.

Teach-First or Write-First

Comparison of major steps for teach-first and write-first.

Teach-First	Write-First
transform tool	write & compile program
teach locations	send program
write & compile program	transform tool
send program	teach locations
run application	run application

CHAPTER 2

Teach-First Method

The teach-first method is useful for working out your application in the workcell, moving the arm from one location to the next as you determine the order of robot work. Once you have decided on the sequence of motion, you can write your program. Using the teach-first method requires that you have a robot, preferably with the end-of-arm tooling (such as a gripper), to teach locations.

Tasks and Recommended Tools

You can develop a simple robot application using the following sequence of tasks with preferred tools or alternative tools.

Preferred Tools

The procedures on the following pages prefer these tools.

Tool Task	Robcomm3	ash	Teach Pendant
1. Open		**open application**	
2. Transform		**set tool transform**	
3. Teach		**create location,** **teach location,** **check location**	**move arm**
4. Set		**create other variables,** **set values**	
5. Write	**write program,** **compile program**		
6. Send	**send program to controller**		
7. Run		**run complete application**	

Alternative Tools

If you want, or if your tool selection is limited, you can use either the preferred tool (in **bold**) or the alternative tool.

Tool Task	Robcomm3	ash	Teach Pendant
1. Open		**open application**	open application
2. Transform		**set tool transform**	
3. Teach		**create location,** move arm, **teach location,** **check location**	create location, **move arm,** teach location, check location
4. Set		**create other variables,** **set values**	create other variables, set values
5. Write	**write program,** **compile program**		
6. Send	**send program to controller**		
7. Run		**run complete application**	run complete application

Task 1 Start Up

The application development procedures in this chapter assume that the system and necessary tools are started.

Task 1.1 Start the Computer

Before you begin, make sure that you have properly installed Robcomm3 and CROS-500C on the computer.

To start the computer and computer-based tools:

1. Turn on your computer.
2. Start Robcomm3.
3. If you are using Robcomm3 for the first time, make sure that the communication setting (available from the C500 menu) matches the communication port where you attached the cable between the computer and the controller.
4. Open the terminal window. In Robcomm3, click the Terminal button, or select Terminal from the C500 menu. With the controller not yet powered on, the window is blank.

Task 1.2 Start the Robot

To start the robot system and the robot-based tools:

1. If you have a teach pendant, make sure that it is connected to the controller. If not, remove the over-ride plug and connect the pendant.
2. Power on the controller at the main power switch on the front panel. As the controller starts up, lines describing start up activity are displayed at the terminal.
3. When the controller is finished its start up activity, check that the terminal displays the $ (system shell) prompt. If the $ (system shell) prompt does not appear automatically, press the Enter key.

Depending on the configuration in the controller, the pendant may or may not be enabled.

Task 2 Create or Open an Application

When developing your robot application, you create an application or “app”, a place on the controller where you store your program and locations. You can create this app with either ashor the teach pendant.

When you develop several robot applications, you create a separate app for each one, keeping the program and locations for one application separate from others.

When you create an app, ash or the teach pendant creates a sub-directory in the app directory of CROS. When you teach locations, ash or the teach pendant automatically stores your locations in that sub-directory. When you send your program from the computer to the controller, Robcomm3 automatically sends your program to that sub-directory.

Preferred Tool

ash

Alternative Tool

Teach Pendant

Opening

If your robot system automatically started up the teach pendant, shut down the teach pendant by pressing ESC on the teach pendant keypad until you reach the terminate screen and press F1 to confirm.

To create or open an app using ash:

1. Start ash. At the $ prompt, enter
   ```
   ash
   ```
2. The shell displays the message “Existing applications are:” and lists all existing applications.
3. Open an application.
   - To create a new application and open it:
     a) Enter the name for your application. For example
        ```
        dispense
        my_app
        ```
        The shell displays the message “Application *application_name* not found -- try to create it?”

     b) Enter `y` for yes. ashcreates the new application, displays a number of messages, and displays a prompt with the application name in it, such as
        ```
        dispense>
        my_app>
        ```
   - To open an existing application:
     a) Enter the name of the application. For example
        ```
        dispense
        my_app
        ```
        The application shell displays a message and then a prompt with the application name in it.

You are now ready to set the tool transform in Task 2.

Task 3 Transform the Tool

A tool transform defines the tool center point (TCP). The tool center point is the work-performing point of your end-of-arm tool, such as the center of gripper fingers, the tip of an applicator, the center of a spray head, etc. The tool transform specifies the position of the TCP relative to the center of the tool flange.

A location is a point in space known to the robot. After you set a tool transform, when you teach a location the point in space that is remembered by the robot is the point of the TCP. When the robot moves, it places the TCP at the location, for example, at the center of the nest, at the point on the work-piece where material is applied, etc.

Task 3.1 Determine the Tool Transform

Determining a tool transform involves measuring the position and orientation of the tool center point relative to center of the tool flange surface. See examples and descriptions in *Application Environment*, the next major part of this manual after *Application Development*.

To determine the transform:

1. Determine the point that will be the tool center point, for example: the mid-point between the gripping spots of two fingers or a point a short distance away from the tip of a dispenser.
2. Measure the distances along the X, Y, and Z axes of the tool co-ordinate system. Remember: the tool co-ordinate system for the F3 has Z rising straight off the tool flange and the A465 and A255 have X rising straight off the tool flange.
3. Determine the orientation, the rotations around the three axes, that will be expressed as yaw, pitch, and roll. The design of most tools involves no rotation, or rotation about only one axis. If this is the case, all three, or two of the three, orientation co-ordinates are zero.

After accurately measuring, use ash to set the tool transform.

Task 3.2 Set the Tool Transform

Tool

ash

Alternative Tool

There is no alternative tool.

Setting Directly

To set the transform:

1. Set the tool transform using the tool command followed by your specific tool parameters.
   Example: A servo gripper with microplate fingers, in metric
   ```
   tool  30, 0, 205, 0, 0, 0
   ```

2. Check the transform by entering the tool command with no parameters
```
     tool
```

Setting Indirectly

To set the transform with values that can also be used in the program, put the transform values in an array of floats. Use the array to set the transform in ashfor teaching and to set the transform later when the program runs.

1. Create a new array of six floats named tran
```
     new  %tran[6]
```
   The array is indexed from 0 to 5: tran[0], tran[1], tran[2], tran[3], tran[4], tran[5].

2. Set each element of the array to one parameter of the tool transform according to this order.

Variable	Transform Parameter
tran[0]	X-axis distance
tran[1]	Y-axis distance
tran[2]	Z-axis distance
tran[3]	yaw rotation (around Z axis)
tran[4]	pitch rotation (around Y axis)
tran[5]	roll rotation (around X axis)

   For example, for a servo gripper with microplate fingers, in metric, the parameters are 30, 0, 205, 0, 0, 0.
```
     set  tran[0] =  30
     set  tran[0] =   0
     set  tran[0] = 205
     set  tran[0] =   0
     set  tran[0] =   0
     set  tran[0] =   0
```

3. Check the settings with the print command
```
     print tran
```
   The values are now in the array elements. You now need to get the array elements into the tool transform setting on the system.

4. Set the tool transform using the array
```
     tool  tran[0], tran[1], tran[2], tran[3], tran[4], tran[5]
```

5. Check the transform by entering the tool command with no parameters
```
     tool
```

Task 4 Create and Teach Locations

Points in space become useable when they are taught as locations.

In the Teach-First method, teaching a location involves the following sequence of tasks.

1. Move the arm to a new position.
2. Create a location variable.
3. Teach a value to a location variable.
4. Check the accuracy of the location.

The last task, checking the accuracy of the locations, is optional, but recommended.

Warning! Use safe locations to avoid collisions. *If you cannot easily move the arm directly from one location to the next, teach additional safe locations that are out of the way of obstructions. When you write your program, you can move the arm from one work location to a safe location and then to the next work location.*

Tip: Teaching exact locations can be time-consuming. To quickly develop your application, you can teach approximate locations, perform a test run, and later re-teach the accurate locations.

Task 4.1 Move the Arm to a New Position

Preferred Tool

Teach Pendant

Alternative Tool

ash

Moving

Use the teach pendant in Velocity-Joint motion to move the arm to positions you want to teach. With Velocity type, motion continues while you press the axis key, but stops when you release the key. With Joint mode, you move each joint individually by pressing the axis keys.

1. At the controller front panel, turn on arm power by pressing the button near the upper right corner. The LED in the button lights up.
2. Ensure that the teach pendant is running. If you have only been using ashat the terminal window, start the teach pendant by entering
   ```
   pendant
   ```
   The pendant displays the Application screen with the name of the application on the second line.
3. At the teach pendant, select Edit by pressing the F1 key.
4. Select Motion by pressing F3. The Manual Menu displays with Motion Type (motn) available at F3 and Motion Mode (mode) available at F4.
5. Scroll through motion types by pressing F3. If you are a new user, select Velocity (vel) for constant motion or Jog for incremental motion.

6. Scroll through motion modes by pressing F4. If you are a new user, select World to move along a world axis, Tool to move along a tool axis, or Joint to move an individual joint.
7. Select a speed by pressing Speed Up and Speed Down. If you are a new user, use a speed of 25% or slower.
8. Move the tool center point to your location using the teach pendant. You can change motion type, motion mode, or speed, as needed.
   - Squeeze the live-man switch on the right side of the teach pendant, using adequate but not excessive pressure.
   - Press an axis key to move in either a positive or negative direction.
   - If arm power shuts off, turn it on by pressing the Arm Power switch on the controller.
9. When the tool center point is at the position you want, you are ready to create a location.

Task 4.2 Create a Location Variable

Preferred Tool

ash

Alternative Tool

Teach Pendant

Creating

Once the arm is positioned in your desired location, you can use the ash New command to create a variable. This variable holds the location data and is used in the application program.

Use a prefix character to specify the type of location variable you want to use.

Prefix Character	Creates
_ (underscore)	a cartesian location (cloc)
# (number sign)	a precision location (ploc)

1. At the application prompt, enter

```
new  prefix_characterlocation_name
```

For example:

```
new  _pick
new  #place
new  _safe_1
```

2. Confirm that your location variable was created by entering

```
list
```

A list of variables created for the application is displayed. Your new location name should be in the list with an asterisk, indicating that it has not been taught.

If you created an incorrect type of variable or used an incorrect name, delete the variable using the erase command.

Task 4.3 Teach a Value to the Location Variable

Preferred Tool

ash

Alternative Tool

Teach Pendant

Teaching

Make sure that the tool center point is at the point you want to teach to the variable.

To teach the location variable:

1. Enter
```
     here  location_variable_name
```
For example
```
     here  pick
     here  place
     here  safe_1
```

2. Confirm that the variable was taught by entering
```
     list
```
The location displays without an asterisk, indicating that it has been taught.

3. Check the positional data for the location variable by entering
```
     print  location_variable_name
```
For example
```
     print  pick
     print  place
```
The location data displays, separated by commas and enclosed within brackets.

If “unknown” is displayed, the data was not taught to the variable.

Task 4.4 Check the Location

Preferred Tool

ash

Alternative Tool

Teach Pendant

Checking

Once you have taught a location, you can check the accuracy of your locations by moving the robot to the location. Although recommended, this procedure is optional.

Warning! Start with a slow speed (speed 10) and be prepared to strike an e-stop button to prevent a possible collision. *When you enter a motion command, the arm moves to the location and expects no obstructions in the way.*

1. Set a slow speed. Enter
```
     speed 10
```

for 10% of full speed. The speed can be set from 1% to 100% of full speed.

2. Move the arm to a position away from your location, such as the ready position by entering
   ```
   ready
   ```
   or another suitable position where there is no possibility of collision when the arm moves from the position to your location.

3. Approach the location with the appro command. The appro command positions the arm near the location, but away from it at the distance that you specify. Enter
   `appro` *`location_variable_name`*, *`distance`*

   For example:
   ```
   appro  pick, 4
   appro  place,6
   ```
   To place the arm closer to the location, you can specify a smaller distance.

4. Move to the location with the move command. Enter
   `move` *`location_variable_name`*

   For example:
   ```
   move  pick
   move  place
   move  safe_1
   ```

5. Depart from the location with the depart command. The depart command positions the arm away from the location by the distance that you specify. Enter
   ```
   depart  distance
   ```
   For example:
   ```
   depart  4
   ```

Repeat

Repeat the teaching tasks for each location.

Task 5 Set Non-Location Variables

Types of non-location variables include:

- integer (a whole number, such as 0, 1, 9., 863)
- float (a number with a decimal point, such as 6.25, 1.755, 99.99)
- string (a group of characters, such as "Press F1 to start."

Non-location variables can also be teachable. You can change their values outside the program, the same way that you can change a teachable location outside a program.

You can use teachable non-location variables to:

- set the number of re-tries of a particular operation
- change the value used in a calculation for the type of material used in the application.
- modify a message sent to the terminal or teach pendant screen at a point in the program

Task 5.1 Create a Non-Location Variable

Preferred Tool

ash

Alternative Tool

Teach Pendant

Creating

When you create a new teachable variable with ashyou use the new command.

When you create the variable, you must specify the type of teachable variable with a prefix character. Creating a variable without a prefix character creates it as an integer.

Prefix Character	Creates
(no prefix character)	integer number (int)
% (percentage sign)	floating point number (float)
$ (dollar sign)	string of characters (string)

1. At the application prompt, enter
   ```
   new  prefix_characterlocation_name
   ```
   For example:
   ```
   new   cycle_times
   new  %analysis_factor
   new  $error_message_5
   ```

2. Confirm that the non-location variable was created by entering
   ```
   list
   ```
   Your new variable name, if created, is displayed with an asterisk, indicating that you have not yet taught it (set a value to it).

If you created an incorrect type of variable or used an incorrect name, use the erase command to delete it, and start over.

Task 5.2 Set a Value to the Non-Location Variable

Preferred Tool

ash

Alternative Tool

Teach Pendant

Setting

Once a variable exists, you can assign it a value or “set a value to the variable”. When you set a value to a variable with ash, you use the set command.

A value can either be a constant or another variable. Certain characters can only be used with different types of variables, as shown in the following table.

Variable Type	Allowable Value
int	Positive or negative whole number. If unsigned, it is taken as positive. (+5, -9, 7382, -9418)
float	Positive or negative number with a decimal point. If unsigned, it is taken as positive. (+2.75, -3.333, 827.0, -0.99)
string	Letters (A, B, C, …, Z, a, b, c, …, z), numerals (0, 1, 2, …, 9), blankspace, punctuation but not \ [the backslash character]. Characters for a string are written between double quotes (“message”). A string is a specific size and takes only as many characters as its size allows. Extra characters are lost.

1. To review the names of variables, enter
   ```
   list
   ```
2. At the application prompt, enter
   ```
   set  variable name  =  value
   ```
   For example:
   ```
   set  cycle_times = 125
   set  x_increment  =  1.33333
   set  error_message_5 = "Waiting for input."
   set  number_of_loops = number_of_samples
   ```
3. Use the print command to check that your variable was properly set. Enter
   ```
   print  variable name
   ```
   For example:
   ```
   print  cycle_times
   ```

Task 6 Write Your RAPL-3 Program

Preferred Tool

Robcomm3

Alternative Tool

Plain text editor and command-line compiler

Writing

After teaching your locations, you can write and compile your RAPL-3 program using Robcomm3. Using specific commands and determining the structure of a program is an extensive subject. Refer to the *RAPL-3 Language Reference Guide* for details.

In summary, to write a RAPL-3 program:

1. In Robcomm3, open an editor window by selecting File|New.
2. Save your program, using the Save As function, to the directory you want. Use an appropriate file name, such as the same name as your application. Make sure the extension is .r3.
3. Setup an Application. From Application, select New. Name your app using the same name as your program file.
4. Check the settings in the App Setup. From Application, select Setup. Make sure that the source file name is the same as your program, the object file name is the same, and the directory name on the controller is the same as the one you used when you taught your locations.
5. In the Editor window, type your program.
6. Compile your program (from source file to object file).
7. Continue adding material to your program. Continue de-bugging your program after attempts to compile.

RAPL 3 Programming Tips

To ensure that your RAPL-3 program integrates properly into your application:

Keep file names the same

With standard applications, use the same name for: the application, the program in source format and compiled object format, and the variable file. For example, if, at the terminal, you created an application named “dispense”, save your program file as “dispense.r3”. When you set up to compile, use the same name. On the controller, when you taught your locations, ash automatically placed the variables in a file named “dispense.v3”.

Location **Computer** **Controller**

Entity		
application (name in ash, sub-directory in \app)		dispense
application (name in Robcomm New App)	dispense.app	
source file (program in readable RAPL-3)	dispense.r3	
object file (program in compiled form)	dispense.	dispense.
variable file (file with teachables)		dispense.v3

Use the same variable names

When you write locations into your program, ensure that you use the same names that you used when you created and taught them in ash. The names must match or the taught values will not be used in the program.

Note: To display all the location and other teachable variable names in your application, use the list command.

Make your variables teachable

For variables that you taught in ashdeclare these variables in your program as “teachable”. This allows your program to use values in the variable (.v3) file that are external to the program. If a variable is not declared as teachable, its value can exist only within the program.

To declare a teachable variable, precede the variable name with "teachable" along with type of variable (cloc, ploc, int, float, string). For a string, also include the length of the string in square brackets. For example:

```
     teachable  cloc  pick, place, safe_1, safe_2
     teachable  int  cycles
     teachable  string[64]  message_wait, message_go
```

***Warning! Use safe locations to avoid collisions.** If you cannot move the arm directly from one location to the next without obstruction, then you must include additional safe locations in your program out of the way of obstructions. Your program can move the arm from one work location to a safe location and then to the next work location.*

Include a tool transform

In earlier tasks, you set a tool transform and then taught locations. As a result, the locations stored in the controller are locations for the tool center point, not for the tool flange of the arm. If you do not include a tool transform in your program, you will have serious collisions. The controller will start up. There will be no transform set. The robot will move to place the arm’s tool flange at the location. This will drive the tool through the nest or work-piece.

If you created an array of floats to use when teaching in ash and to use in your program when running, write the array into the tool_set() command. For example, tool_set(tran)

Task 7 Send the Program to the Controller

Tool

Robcomm3

Alternative Tool

There is no alternative tool.

Sending

You have written and compiled your program on the computer. You must send it to the controller, to use it to run the robot.

With Robcomm3, the procedure for transferring your program to the controller is simplified if you use the App Setup feature described in the previous task.

1. Send the program using the Send function available from the Send button, Application menu, or right-mouse click menu. The Send File Transfer Progress window displays the transfer.
2. Check that your program was successfully transferred to the appropriate place. In the terminal window, with ash running and your application open, display the contents with the dir command.

Task 8 Run Your Application

Preferred Tool

ash

Alternative Tool

Teach Pendant

Running

To run your application using ash:

1. At the controller front panel, switch Arm Power on. The LED in the switch button lights up.
2. Move the arm to a safe position.

   Warning! Be prepared to strike an emergency-stop button to stop arm motion. *The program moves the arm directly from one location to the next and expects no obstructions.*

3. At the application prompt, enter

   ```
   run
   ```

   Using the run command without any parameters runs the program file and variable file with the same name as the open application.

CHAPTER 2

Write-First Method

Use the Write-First method of application development if you are comfortable with writing programs, or if you do not have a robot available to teach the robot locations first.

Using this method, you develop your application by working out the flow of robot tasks as you write the program and name locations. After you compile and send your program to the controller, a utility in ash automatically creates a variable file containing all teachable variables with the names you used in your program. Then, you can teach the locations with the robot arm.

Tasks and Recommended Tools

You can develop a simple robot application using the following sequence of tasks with preferred tools or alternative tools.

Preferred Tools

The procedures on the following pages prefer these tools.

Tool Task	Robcomm3	ash	Teach Pendant
1. Write	**write program, compile program**		
2. Send	**send program to controller**		
3. Open		**open application**	
4. Transform		**set tool transform**	
5. Teach		**teach location, check location**	**move arm**
6. Set		**set values to other variables**	
7. Run		**run application**	

Alternative Tools

If you want, or if your tool selection is limited, you can use either the preferred tool (in **bold**) or the alternative tool.

Tool Task	Robcomm3	ash	Teach Pendant
1. Write	**write program, compile program**		
2. Send	**send program to controller**		
3. Open		**open application**	open application
4. Transform		**set tool transform**	
5. Teach		**review locations** move arm, **teach location, check location**	review locations **move arm,** teach location, check location
6. Set		**set values to other variables**	set values to other variables
7. Run		**run application**	run application

Task 1 Start Up

The application development procedures in this chapter assume that the system and necessary tools are started.

Task 1.1 Start the Computer

Before you begin, make sure that you have properly installed Robcomm3 and CROS-500C on the computer.

To start the computer and computer-based tools:

1. Turn on your computer.
2. Start Robcomm3. (If you are away from the robot, or are going to spend considerable time writing and de-bugging your program, stop here, complete the writing task, and when you are ready for the remaining tasks, continue this start up.)
3. If you are using Robcomm3 for the first time, make sure that the communication setting (available from the C500 menu) matches the communication port where you attached the cable between the computer and the controller.
4. Open the terminal window. In Robcomm3, click the Terminal button, or select Terminal from the C500 menu. With the controller not yet powered on, the window is blank.

Task 1.2 Start the Robot

To start the robot system and the robot-based tools:

1. If you have a teach pendant, make sure that it is connected to the controller. If not, remove the over-ride plug and connect the pendant.
2. Power on the controller at the main power switch on the front panel. As the controller starts up, lines describing start up activity are displayed at the terminal.
3. When the controller is finished its start up activity, check that the terminal displays the $ (system shell) prompt. If the $ (system shell) prompt does not appear automatically, press the Enter key.

Depending on the configuration in the controller, the pendant may or may not be enabled.

Task 2 Write Your RAPL-3 Program

Preferred Tool

Robcomm3

Alternative Tool

Plain text editor and command-line compiler

Writing

Write and compile your RAPL-3 program using Robcomm3. Using specific commands and determining the structure of a program is an extensive subject. Refer to the *RAPL-3 Language Reference Guide* for details.

In summary, to write a RAPL-3 program:

1. In Robcomm3, open an editor window by selecting File|New.
2. Save your program, using the Save As function, to the directory you want. Use an appropriate file name, such as the same name as your application. Make sure the extension is .r3.
3. Setup an Application. From Application, select New. Name your app using the same name as your program file.
4. Check the settings in the App Setup by selecting Setup from Application. Make sure that the source file name is the same as your program and the object file name is the same.
5. In the Editor window, type your program.
6. Compile your program (from source file to object file).
7. Continue adding material to your program. Continue de-bugging your program after attempts to compile.

RAPL 3 Programming Tips

To ensure that your RAPL-3 program integrates properly into your application:

Keep file names the same

With standard applications, use the same name for: the program in source format and compiled object format and for the app.

Entity	Name
application (name in Robcomm New App)	dispense.app
source file (program in readable RAPL-3)	dispense.r3
object file (program in compiled form)	dispense.

Make your variables teachable

For variables that you want to teach on the controller (usually all locations and some non-location variables) declare these variables as “teachable”. This allows your program to use values stored on the controller. If a variable is not declared as teachable, its value can exist only within the program.

To declare a teachable variable, precede the variable name with "teachable" along with type of variable (cloc, ploc, int, float, string). For a string, also include the length of the string in square brackets. For example:

```
     teachable  cloc  pick, place, safe_1, safe_2
     teachable  int  cycles
     teachable  string[64]  message_wait, message_go
```

Use safe locations

If the arm will not be able to move directly from one location to the next without obstruction, include additional safe locations, out of the way of obstructions. Your program can move the arm from one work location to a safe location and then to the next work location.

Include a tool transform

You have options.

- Determine the tool transform detailed in Transforming the Tool. In the program, use these constants (“hard-code” the parameters) in the tool_set() command.
- In the program, declare a teachable array of floats and use the array in the tool_set() command. Work out the other details when you are teaching locations with the arm. Determine the tool transform. Using those parameters, set values to the array of floats that will be used when the program runs. Using the same parameters, set a tool transform in ash that will be used when teaching.

Task 3 Send the Program to the Controller

Tool

Robcomm3

Alternative Tool

There is no alternative tool.

Sending

You have written and compiled your program on the computer. You must send it to the controller.

With Robcomm3, the procedure for transferring your program to the controller is simplified if you use the App Setup feature.

Task 3.1 Set Up the App

When you set up the app for compiling from source file to object file, you may have set up for sending from the computer to the controller.

1. Make sure that the app is open.
    - If you have been writing and compiling, you likely have the app open. From the Application menu, New App and Open App are disabled. Check which app is open by selecting Setup. If the wrong app is open, close it by selecting Close App.
    - If you do not have the app open, or re-started Robcomm3 since writing and compiling, open the app. From the Application menu, select Open App. At the selection window, select the proper app.
2. Check the settings in the App Setup. From Application, select Setup. Make sure that the sub-directory on the controller is where you want to send the program. With a standard application, the sub-directory name has the same name as the program file.

Task 3.2 Send Your Program to the Controller

Details of sending the program are handled automatically if you have set up the app. Otherwise you have to use the File Transfer utility.

1. Send the program using the Send function available from the Send button, the Application menu, or the right-mouse click menu.
2. Check that your program was successfully transferred to the appropriate place. In the terminal window, from the system shell, use the dir command
    ```
    dir  \app\application_name
    ```
    for example
    ```
    dir  \app\dispense
    ```
    The program file in compiled object format does not have a .r3 extension.

Task 4 Open the Application

When developing your robot application, you use an application or "app", a place on the controller where you store your program and locations.

When you develop several robot applications, you use a separate app for each one, keeping the program and locations for one application separate from others.

An app is really a sub-directory in the app directory of CROS. When you sent your program from the computer to the controller, Robcomm3 automatically sent it to that sub-directory. When you teach locations, ash or the teach pendant automatically stores your locations in that sub-directory.

Preferred Tool

ash

Alternative Tool

Teach Pendant

Opening

If you sent the program automatically with the App Setup feature, the application on the controller was automatically created. You only have to open it.

1. Startash. At the $ prompt, enter
   ```
   ash
   ```
2. The shell displays the message "Existing applications are:" and lists all existing applications.
3. Open the application.
   - If the application that you want already exists:
     a) Enter the name of the application. For example
        ```
        dispense
        my_app
        ```
        The application shell displays a message and then a prompt with the application name in it.
   - To create a new application and open it:
     a) Enter the name for your application. For example
        ```
        dispense
        my_app
        ```
        The shell displays the message "Application *application_name* not found -- try to create it?"

     b) Enter `y` for yes. Ash creates the new application, displays a number of messages, and displays a prompt with the application name in it, such as
        ```
        dispense>
        my_app>
        ```
     c) Search for your program file. Since the application did not already exist when you started ash, the application was not created in the proper place and your file did not go to the proper place when you sent the file. Either use system shell

commands to search directories and move your program to this app directory, or re-send the file after re-checking the app setup in Robcomm3.

You are now ready to set the tool transform in Task 2.

Task 5 Transform the Tool

A tool transform defines the tool center point (TCP). The tool center point is the work-performing point of your end-of-arm tool, such as the center of gripper fingers, the tip of an applicator, the center of a spray head, etc. The tool transform specifies the position of the TCP relative to the center of the tool flange.

A location is a point in space known to the robot. After you set a tool transform, when you teach a location the point in space that is remembered by the robot is the point of the TCP. When the robot moves, it places the TCP at the location, for example, at the center of the nest, at the point on the work-piece where material is applied, etc.

5.1 Determine the Tool Transform

Determining a tool transform involves measuring the position and orientation of the tool center point relative to center of the tool flange surface. See examples and descriptions in *Application Environment*.

To determine the transform:

1. Determine the point that will be the tool center point, for example: the mid-point between the gripping spots of two fingers or a point a short distance away from the tip of a dispenser.
2. Measure the distances along the X, Y, and Z axes of the tool co-ordinate system. Remember: the tool co-ordinate system for the F3 has Z rising straight off the tool flange and the A465 and A255 have X rising straight off the tool flange.
3. Determine the orientation, the rotations around the three axes, that will be expressed as yaw, pitch, and roll. The design of most tools involves no rotation, or rotation about only one axis. If this is the case, all three, or two of the three, orientation co-ordinates are zero.

After accurately measuring, use ash to set the tool transform.

5.2 Set the Tool Transform

Tool

ash

Alternative Tool

There is no alternative tool.

Setting Directly

To set the transform:

1. Set the tool transform using the tool command followed by your specific tool parameters.
   Example: A servo gripper with microplate fingers, in metric
   ```
   tool  30, 0, 205, 0, 0, 0
   ```

2. Check the transform by entering the tool command with no parameters
```
     tool
```

Setting Indirectly

To set the transform with values that can also be used in the program, put the transform values in an array of floats. Use the array to set the transform in ash for teaching and to set the transform later when the program runs.

1. Create a new array of six floats named tran
```
     new  %tran[6]
```
   The array is indexed from 0 to 5: tran[0], tran[1], tran[2], tran[3], tran[4], tran[5].

2. Set each element of the array to one parameter of the tool transform according to this order.

Variable	Transform Parameter
tran[0]	X-axis distance
tran[1]	Y-axis distance
tran[2]	Z-axis distance
tran[3]	yaw rotation (around Z axis)
tran[4]	pitch rotation (around Y axis)
tran[5]	roll rotation (around X axis)

   For example, for a servo gripper with microplate fingers, in metric, the parameters are 30, 0, 205, 0, 0, 0.
```
     set  tran[0] =  30
     set  tran[0] =   0
     set  tran[0] = 205
     set  tran[0] =   0
     set  tran[0] =   0
     set  tran[0] =   0
```

3. Check the settings with the print command
```
     print tran
```
   The values are now in the array elements. You now need to get the array elements into the tool transform setting on the system.

4. Set the tool transform using the array
```
     tool  tran[0], tran[1], tran[2], tran[3], tran[4], tran[5]
```

5. Check the transform by entering the tool command with no parameters
```
     tool
```

Task 6 Teach Locations

When you wrote your program, you declared some teachable locations. You sent the program to the controller. A utility of ash adds the teachables into the variable file and loads them into the ash database.

In the Write-First method, teaching a location involves the following sequence of tasks.

1. Review your location variables.
2. Move the arm to a new position.
3. Teach the value into the location variable.
4. Check the accuracy of the location.

The last task is optional, but recommended.

Task 6.1 Review Your Location Variables

Tool

ashAlternative Tool

Teach pendant

Reviewing

When you opened the application (when you started ash with the ash command at the terminal) the teachable variables from your program were copied into a variable file and loaded into the ash database. Check that this happened. If not, there are corrective steps.

1. At the ash prompt (the name of your application with an angle bracket) such as
   ```
   application_name>
   ```
   list the teachable variables in the database with the list command
   ```
   list
   ```
   The variables from the database are displayed at the terminal.
2. Compare the list of variables to your knowledge of the program.

Troubleshooting

If there are no variables listed, the utility that generates the variables may not have run successfully. Run it now to generate the variables from the program using the refresh command.

1. At the prompt, enter
   ```
   refresh
   ```
   A line for each variable (added from the program to the variable file) is displayed.
2. Check that they are in the database by entering
   ```
   list
   ```
   The variables should be displayed.

If the refresh command did not generate variables, the program file may not be in the directory.

1. Search for your program file, using system shell commands. List all

contents of all directories (with the –Recursive flag), starting from the root directory (/), by entering

```
dir -R /
```

All directories, sub-directories, etc. down to each file, are listed.

2. Look for your file by name. If you locate it, more it with the file move command

```
mv current_file_location preferred_file_location
```

for example

```
mv /dispense /app/dispense/dispense
```

If the file is not on the controller, you need to send it.

1. In Robcomm3, check that the Open App feature is open and that the App Setup configuration is set to send your object file to the proper application directory on the controller, for example

```
/app/dispense
/app/spray
```

2. From Robcomm3, send your file with the Send feature available from the right-mouse click menu, the Send button, or Send command from the drop-down menu.
3. Check that your file is in the proper app directory on the controller. In the terminal window, at the application prompt enter

```
dir
```

4. At the prompt, enter

```
refresh
```

A line for each variable (added from the program to the variable file) is displayed.

5. Check that they are in the database by entering

```
list
```

The variables should be displayed.

Task 6.2 Move the Arm and Teach the Location

Tool

Teach pendant

Alternative Tool

ash

Moving and Teaching

To teach a variable from the teach pendant, you select the variable you want to teach, move the robot to the position you want, and then enter the teach command. Use the teach pendant in "Joint-Velocity" mode to move the arm to a position. In joint mode you move each joint individually by pressing the axis keys. Velocity motion means that the axis continues to move but stops when you release the key.

Before you begin:

- Pass point of control to the teach pendant by entering the pendant command at the terminal.

- Ensure the application you created is active at the teach pendant.
- Ensure that the robot is homed.

To teach a location variable:

1. Select the variable you want to teach. At the variable find screen, locate and select the location variable you want to teach, and press the F1(sel) key to select the variable. The manual menu screen opens.
2. Press F4 repeatedly until you select Joint mode.
3. Press F3 repeatedly until you select Velocity motion.
4. Select a speed, press Speed Up or Speed Down.

   Tip: New users should use a speed of 25% or slower.
5. At the controller front panel, turn on Arm Power. The LED in the switch button lights up.
6. Squeeze the live-man switch on the right side of the teach pendant, using adequate but not excessive pressure.
   a) Press an axis key to move a joint in either its positive or negative direction.
   b) Release the key to stop arm motion.
   c) If arm power shuts off, turn it on by pressing the arm power switch.

   Note: The live-man switch on the teach pendant must be squeezed in order to move the robot.
7. To move the arm to your desired location:
   a) Move joints 1, 2, or 3 to position the arm near the location.
   b) Move wrist joints 4, 5, and 6 for wrist orientation.
   c) Continue to move the arm with the different axis keys until the arm is at the position you want.

   Tip: Teaching exact locations can be time-consuming for new users. Teach locations roughly, perform a test run, and then later re-teach the locations exactly.
8. When the robot is positioned, press the F1(tch) key to teach the location.
9. If you teach the data to the wrong location variable, position the arm and then re-teach the location. New location data replaces the existing data in the variable.

Task 6.3 Check the Accuracy of Your Locations

Tool

ash

Alternative Tool

Teach pendant

Once you have taught and saved several locations, check the accuracy of your locations at different speeds. Although recommended, this procedure is optional.

Warning! Start with a slow speed and be prepared to strike an e-stop button to prevent a possible collision. *When you enter a motion command, the arm moves to the location and expects no obstruction.*

1. Set a slow speed, enter `speed 10`.

   Note: The speed ranges from 1% to 100% of full speed.

2. Use the following procedure to check the accuracy of each of your locations:

   a) Use the appro command to approach the location. The appro command positions the arm near the location, at the distance you specify.

      Enter
      ```
         appro  location variable name , distance
      ```
      For example:
      ```
         appro  pick,4
         appro  place,6
      ```

      Note: To place the arm closer to the location, you can specify a smaller number.

   b) Use the Move command to position the arm at an exact location.

      Enter
      ```
         move  location variable name
      ```
      For example:
      ```
         move  pick
         move  place
         move  safe_1
      ```

   c) Use the depart command to depart from the location. The depart command positions the arm away from the location at a distance you specify.

      Enter
      ```
         depart  distance
      ```
      For example:
      ```
         depart  4
      ```

      Note: To use the depart command, the arm must be at a location, not at a depart or appro position.

3. If the arm is moving without obstruction, increase the speed in step 1 and repeat step 2 until you find an appropriate speed in which to run your application.

Task 7 Set Teachable Non-Location Variables

If you declared teachable non-location variables in your program, use the following procedure to set a value to a variable.

Task 7.1 Set Your Non-Location Variables

Tool

ash

Alternative Tool

Teach pendant

Setting

Once a variable exists, you can assign it a value or “set a value to the variable”. When you set a value to a variable with ash, you use the set command.

A value can either be a constant or another variable. Certain characters can only be used with different types of variables, as shown in the following table.

Variable Type	Allowable Value
int	Positive or negative whole number. If unsigned, it is taken as positive. (+5, -9, 7382, -9418)
float	Positive or negative number with a decimal point. If unsigned, it is taken as positive. (+2.75, -3.333, 827.0, -0.99)
string	Letters (A, B, C, …, Z, a, b, c, …, z), numerals (0, 1, 2, …, 9), blankspace, punctuation but not \ [the backslash character]. Characters for a string are written between double quotes (“message”). A string is a specific size and takes only as many characters as its size allows. Extra characters are lost.

4. To review the names of variables, enter
```
     list
```
5. At the application prompt, enter
```
     set  variable name  =  value
```
   For example:
```
     set  cycle_times = 125
     set  x_increment  =  1.33333
     set  error_message_5 = "Waiting for input."
     set  number_of_loops = number_of_samples
```
6. Use the print command to check that your variable was properly set. Enter
```
     print  variable name
```
   For example:
```
     print  cycle_times
```

Task 8 Run Your Application

Preferred Tool

ash

Alternative Tool

Teach Pendant

Running

When you run your application, the controller executes your program using the variable file containing your taught locations.

To run your application using ash:

1. At the controller front panel, switch Arm Power on. The LED in the switch button lights up.
2. Move the arm to a safe position.

   Warning! Be prepared to strike an emergency-stop button to stop arm motion. *The program moves the arm directly from one location to the next and expects no obstructions.*

3. At the application prompt, enter

   ```
       run
   ```

   Using the run command without any parameters runs the program file and variable file with the same name as the open application.